



Fundamental of SOFTWARE ENGINEERING & Digital Transformation



Software Engineering :
An Important Role in
Digital Transformation

อธิบายกระบวนการ Digital Transformation
ด้วย Software Engineering ถ่ายทอดจากประสบการณ์
ในการพัฒนาระบบเทคโนโลยีดิจิทัลให้กับหน่วยงาน
ทั้งภาครัฐและเอกชน

ผู้แต่ง สร. ดร.ปานใจ สารทัศนวงศ์
บรรณาธิการ กิรพล ชาญเจริญ

CONTENT

CHAPTER

1

การปรับเปลี่ยนสู่ดิจิทัล DIGITAL TRANSFORMATION 1

- 1.1 เศรษฐกิจดิจิทัล (Digital Economy) 2
- 1.2 ความหมายของเทคโนโลยีดิจิทัล (Digital Technology Definition) 3
- 1.3 องค์กรดิจิทัล (Digital Enterprises) 5
- 1.4 การปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation) 6
 - 1.4.1 ขั้นตอนการปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation Roadmap) 6
 - 1.4.2 กรอบการปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation Framework) 8
- 1.5 บทบาทของวิศวกรซอฟต์แวร์ในการปรับเปลี่ยนสู่ดิจิทัล (The Role of Software Engineer in Digital Transformation) 12
 - 1.5.1 ความสำคัญของวิศวกรซอฟต์แวร์ 12
 - 1.5.2 ความท้าทายของการปรับเปลี่ยนสู่ดิจิทัลในวิศวกรซอฟต์แวร์ 13
 - 1.5.3 แนวทางการเอาชนะความท้าทายของการปรับเปลี่ยนสู่ดิจิทัลในด้านวิศวกรซอฟต์แวร์ 15
- 1.6 คำนิยามและคำจำกัดความที่ใช้ในวิศวกรรมซอฟต์แวร์ 17

CHAPTER

2

การวิเคราะห์เชิงโครงสร้าง STRUCTURED ANALYSIS 23

- 2.1 การวิเคราะห์เชิงโครงสร้าง (Structure Analysis And Design) 24
- 2.2 แผนภาพการไหลของข้อมูล (Data Flow Diagram) 25
 - 2.2.1 แผนภาพบริบท (Context Diagram) 26
 - 2.2.2 แผนภาพการไหลของข้อมูลระดับที่ 0 (DFD Level 0) 27
 - 2.2.3 การแยกองค์ประกอบ (Decomposition) 28
- 2.3 ตัวอย่างการใช้แผนภาพการไหลของข้อมูลกับการปรับหรือโครงสร้างทางธุรกิจ (Using DFD in BPR) 30

CHAPTER

3

การวิเคราะห์และออกแบบเชิงวัตถุ OBJECT-ORIENTED ANALYSIS AND DESIGN 33

- 3.1 กระบวนทัศน์เชิงวัตถุ (Object-Oriented Paradigm) 34
- 3.2 หลักการเชิงวัตถุ (Object-Oriented Concept) 36
- 3.3 นามธรรม (Abstraction) 37
 - 3.3.1 คลาส (Class) และออบเจกต์ (Object) 38

CONTENT

- 3.3.2 การสื่อสารของออบเจกต์ (Object Communication) 40
- 3.3.3 ปฏิสัมพันธ์ระหว่างออบเจกต์ (Interacting Objects) 41
- 3.3.4 การสื่อสารของออบเจกต์ (Object Communication) 42
- 3.4 การห่อหุ้ม (Encapsulation) 42
- 3.5 ความเป็นโมดูลาร์ (Modularity) 43
- 3.6 ลำดับชั้น (Hierarchy) 43
 - 3.6.1 ลำดับชั้นแบบ IS-A 43
 - 3.6.2 ลำดับชั้นแบบ PART-OF 46
- 3.7 โพลิมอร์ฟิซึม (Polymorphism) 47
- 3.8 การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design) 47
 - 3.8.1 บริบทของระบบและรูปแบบการใช้งาน 48
 - 3.8.2 การออกแบบสถาปัตยกรรม 49
 - 3.8.3 การระบุออบเจกต์และคลาส 49
 - 3.8.4 แบบจำลองการออกแบบ 49
 - 3.8.5 กำหนดส่วนต่อประสานของออบเจกต์ 50

CHAPTER

4

UML ภาษาการสร้างแบบจำลองแบบครบวงจร

53

- 4.1 ภาษาการสร้างแบบจำลอง (Modeling Language) 54

- 4.2 ความเป็นมาของภาษาการสร้างแบบจำลองแบบครบวงจร (Unified Modeling Language, UML) 55
- 4.3 แผนภาพยูเอ็มแอล (UML Diagram) 62
- 4.4 แผนภาพยูสเคส (Use Case Diagrams) 63
 - 4.4.1 องค์ประกอบของยูสเคส (Use Case) 63
 - 4.4.2 ความสัมพันธ์ระหว่างยูสเคส 64
 - 4.4.3 หลักการวิเคราะห์ระบบด้วยยูสเคส 65
- 4.5 แผนภาพคลาส (Class Diagrams) 68
 - 4.5.1 การวิเคราะห์ความทนทาน (Robustness Analysis) 74
 - 4.5.2 ขั้นตอนในการทำแผนภาพคลาส 76
 - 4.5.3 แนวปฏิบัติที่ดีของการทำแผนภาพคลาส 77
- 4.6 แผนภาพลำดับ (Sequence Diagrams) 77
 - 4.6.1 ขั้นตอนในการเขียนแผนภาพลำดับ 81
 - 4.6.2 แนวทางปฏิบัติที่ดีของแผนภาพลำดับ 84
 - 4.6.3 กระบวนการสร้างแผนภาพลำดับด้วยการวิเคราะห์ความคงทน 86
- 4.7 แผนภาพสถานะ (State Diagrams) 87
 - 4.7.1 ขั้นตอนการเขียนแผนภาพสถานะ 89
- 4.8 แผนภาพกิจกรรม (Activity Diagrams) 91
 - 4.8.1 ขั้นตอนการทำแผนภาพกิจกรรม 92
 - 4.8.2 ตัวอย่างแผนภาพกิจกรรม 94
- 4.9 แผนภาพการนำไปใช้ (Deployment Diagrams) 95
 - 4.9.1 ขั้นตอนการทำแผนภาพการนำไปใช้ 96

4.10 การสร้างรายงานโดยยูเอ็มแอล..... 98
 4.11 การพัฒนาโปรแกรมโดยใช้แบบจำลอง..... 100
 4.12 การพัฒนาด้วยวิศวกรรมย้อนกลับ
 (Reverse Engineering) 102

CHAPTER

5

กระบวนการซอฟต์แวร์ SOFTWARE PROCESS 107

5.1 แบบจำลองกระบวนการซอฟต์แวร์ (Software Process Models)..... 110
 5.1.1 แบบจำลองการพัฒนาระบบแบบน้ำตก
 (Water Fall Model) 110
 5.1.2 แบบจำลองการพัฒนาต้นแบบ (Prototyping Model) 114
 5.1.3 แบบจำลองการพัฒนาระบบแบบมีส่วนร่วม
 (Joint Application Development Model)..... 114
 5.1.4 แบบจำลองการพัฒนาซอฟต์แวร์อย่างรวดเร็ว
 (Rapid Application Development Model) 115
 5.1.5 การทำงานแบบบอล์ (Agile Methodology) 116
 5.1.6 แบบจำลองการพัฒนาโปรแกรมแบบเอ็กซ์ตรีม
 (Extreme Programming) 124
 5.1.7 แบบจำลองการพัฒนาโดยวิธีเชิงวัตถุ (Object Oriented Development) 126
 5.1.8 สถาปัตยกรรมเชิงเซอร์วิส (Service Oriented Architecture) 128

5.2 กิจกรรมในกระบวนการซอฟต์แวร์ (Process Activities) 129
 5.2.1 การระบุข้อกำหนดของซอฟต์แวร์ (Software Specification) 130
 5.2.2 การออกแบบซอฟต์แวร์และการนำไปใช้
 (Software Design and Implementation) 132
 5.2.3 การตรวจสอบความถูกต้องของซอฟต์แวร์
 (Software Validation) 134
 5.2.4 วิวัฒนาการของซอฟต์แวร์ (Software Evolution) 135

CHAPTER

6

การจัดการเทคโนโลยีดิจิทัล DIGITAL TECHNOLOGY MANAGEMENT 139

6.1 ปัญหาในการบริหารจัดการองค์กร 141
 6.2 สถาปัตยกรรมองค์กร (Enterprise Architecture) 143
 6.2.1 สถาปัตยกรรมองค์กรในปัจจุบัน (As is Enterprise Architecture)..... 148
 6.2.2 สถาปัตยกรรมองค์กรในอนาคต (To be Enterprise Architecture) 149
 6.3 สถาปัตยกรรมองค์กรกับวิศวกรรมซอฟต์แวร์
 (Enterprise Architecture and Software Engineering)..... 154
 6.4 การกำกับดูแลการใช้เทคโนโลยีดิจิทัล (Digital Technology Governance) 155

CONTENT

6.5 การจัดการซอฟต์แวร์ (Software Management)	159
6.5.1 การจัดการโครงการด้านซอฟต์แวร์ (Software Project Management)	159
6.5.2 การบริหารโครงการซอฟต์แวร์แบบอไจล์ (Agile Project Management)	165
6.5.3 การจัดการความเสี่ยง (Risk Management)	167
6.5.4 การประมาณการต้นทุนซอฟต์แวร์ (Software Cost Estimation)	172
6.6 การจัดการคุณภาพซอฟต์แวร์ (Software Quality Management)	174
6.6.1 การปรับปรุงคุณภาพซอฟต์แวร์ให้มีประสิทธิภาพ	175
6.6.2 CMMI กับกระบวนการพัฒนาแบบอไจล์	177
6.7 ปัจจัยสู่ความสำเร็จ (Keys Success Factors)	178

CHAPTER

7

วิศวกรรมความต้องการ REQUIREMENT ENGINEERING 181

7.1 ความหมายของความต้องการด้านซอฟต์แวร์ (Software Requirement Definition)	183
7.2 กระบวนการวิศวกรรมความต้องการ (Requirement Engineering Process)	185

7.3 กระบวนการจัดทำข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirement Specification Process)	189
7.4 การสกัดและวิเคราะห์ความต้องการ (Requirements Elicitation and Analysis)....	190
7.5 การตรวจสอบความต้องการ (Requirements Validation)	222
7.6 การจัดการความต้องการ (Requirements Management)	223
7.7 วิศวกรรมความต้องการแบบอไจล์ (Agile Requirement Engineering)	224

CHAPTER

8

การออกแบบระบบ SYSTEM DESIGN 231

8.1 กระบวนการออกแบบระบบ (System Design)	234
8.2 การออกแบบสถาปัตยกรรม (Architectural Design)	235
8.2.1 ลักษณะการออกแบบสถาปัตยกรรมที่ดี ...	235
8.2.2 มุมมองการออกแบบสถาปัตยกรรม (Architecture View)	236
8.2.3 รูปแบบสถาปัตยกรรม (Architectural Pattern)	238
8.2.4 สถาปัตยกรรมแอปพลิเคชัน (Application Architecture)	244

8.2.5 การออกแบบสถาปัตยกรรมสำหรับกรณีศึกษา การพัฒนาระบบการออกใบอนุญาตกิจการที่เป็นอันตรายต่อสุขภาพ.....	246	9.2 การทดสอบซอฟต์แวร์สำหรับแบบจำลองน้ำตก (Water Fall Model Software Testing).....	281
8.3 การออกแบบส่วนต่อประสาน (Interface Design)	250	9.2.1 การทดสอบการพัฒนา (Development Testing).....	282
8.3.1 การออกแบบฟอร์มและรายงาน (Designing Forms and Reports).....	250	9.2.2 การทดสอบการเผยแพร่ (Release Testing).....	286
8.4 การออกแบบส่วนประกอบ (Component Design)	255	9.2.3 การทดสอบโดยผู้ใช้งาน (User Testing)	287
8.5 การออกแบบฐานข้อมูล (Database Design)...	257	9.3 การทดสอบซอฟต์แวร์สำหรับไจล์ (Agile Software Testing)	290
8.5.1 ธรรมาภิบาลข้อมูล (Data Governance) ...	258	9.4 การติดตั้งระบบ (System Installation).....	292
8.5.2 การบริหารและกระบวนการจัดการข้อมูลหรือวงจรชีวิตของข้อมูล (Data Life Cycle).....	261	9.4.1 การติดตั้งโดยตรง (Direct installation)....	292
8.5.3 การบูรณาการข้อมูล (Data Integration) ...	262	9.4.2 การติดตั้งแบบขนาน (Parallel Installation).....	293
8.6 รูปแบบการออกแบบ (Design Pattern).....	263	9.4.3 การติดตั้งแบบนำร่อง (Single-Location Installation).....	293
CHAPTER		9.4.4 การติดตั้งเป็นระยะ (Phase Installation) ..	293
9		9.5 การจัดทำเอกสาร (Documentation)	294
การติดตั้งระบบและการบำรุงรักษา		9.6 การฝึกอบรม (Training).....	295
SYSTEM IMPLEMENTATION AND MAINTENANCE	269	9.7 การบำรุงรักษาซอฟต์แวร์ (Software Maintenance).....	295
9.1 การพัฒนาและติดตั้งระบบ (System Implementation)	270	CHAPTER	
9.1.1 การพัฒนาโปรแกรม (Program Coding) ...	270	10	
9.1.2 การพัฒนาด้วยวิศวกรรมย้อนกลับ (Reverse Engineering)	275	วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส SERVICE ORIENTED SOFTWARE ENGINEERING	299
9.1.3 วิธีการพัฒนาโปรแกรมแบบ Agile และ DevOps.....	277	10.1 วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส (Service Oriented Software Engineering).....	300

CONTENT

10.2	รูปแบบการพัฒนาซอฟต์แวร์เชิงเซอร์วิส.....	304
10.3	เทคโนโลยีเว็บเซอร์วิส (Web Services).....	307
10.3.1	เว็บเซอร์วิสแบบ SOAP (SOAP Web Services).....	308
10.3.2	เว็บเซอร์วิสแบบ REST (REST Web Services).....	315
10.3.3	ข้อแตกต่างระหว่างเว็บเซอร์วิสแบบ SOAP และ REST.....	317
10.4	รูปแบบข้อมูลที่ใช้ในเว็บเซอร์วิส.....	317
10.4.1	ภาษา XML	318
10.4.2	JavaScript Object Notation (JSON)....	318
10.4.3	มาตรฐานการแลกเปลี่ยนข้อมูล.....	319
10.5	สถาปัตยกรรมเชิงเซอร์วิส (Service Oriented Architecture, SOA)	330
10.5.1	การประยุกต์ใช้สถาปัตยกรรมเชิงเซอร์วิส (SOA).....	330
10.5.2	ประโยชน์ในการพัฒนาด้วยเทคโนโลยี SOA	330
10.5.3	ขั้นตอนในการพัฒนาสถาปัตยกรรมเชิงเซอร์วิส (SOA).....	331
10.6	ไมโครเซอร์วิส (Microservices).....	337
10.7	ซอฟต์แวร์คอนเทนเนอร์ (Software Container).....	342

CHAPTER

11

การพัฒนาบุคลากรด้านดิจิทัล DIGITAL HUMAN RESOURCE DEVELOPMENTS 347

11.1	ทักษะด้านดิจิทัล	348
11.2	สมรรถนะของบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล	354
11.3	การพัฒนาบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล	355
11.3.1	ระดับประธานฝ่ายสารสนเทศ (Chief Information Officer, CIO)	355
11.3.2	ระดับผู้บริหารโครงการ (Project Management Officer, PMO)	356
11.3.3	ระดับผู้พัฒนาระบบ (Developer)	356
11.3.4	ระดับผู้ดูแลระบบ (System Administrator)	358
11.3.5	ระดับผู้ดูแลโครงสร้างพื้นฐาน (Network/Infrastructure)	359

CHAPTER

1

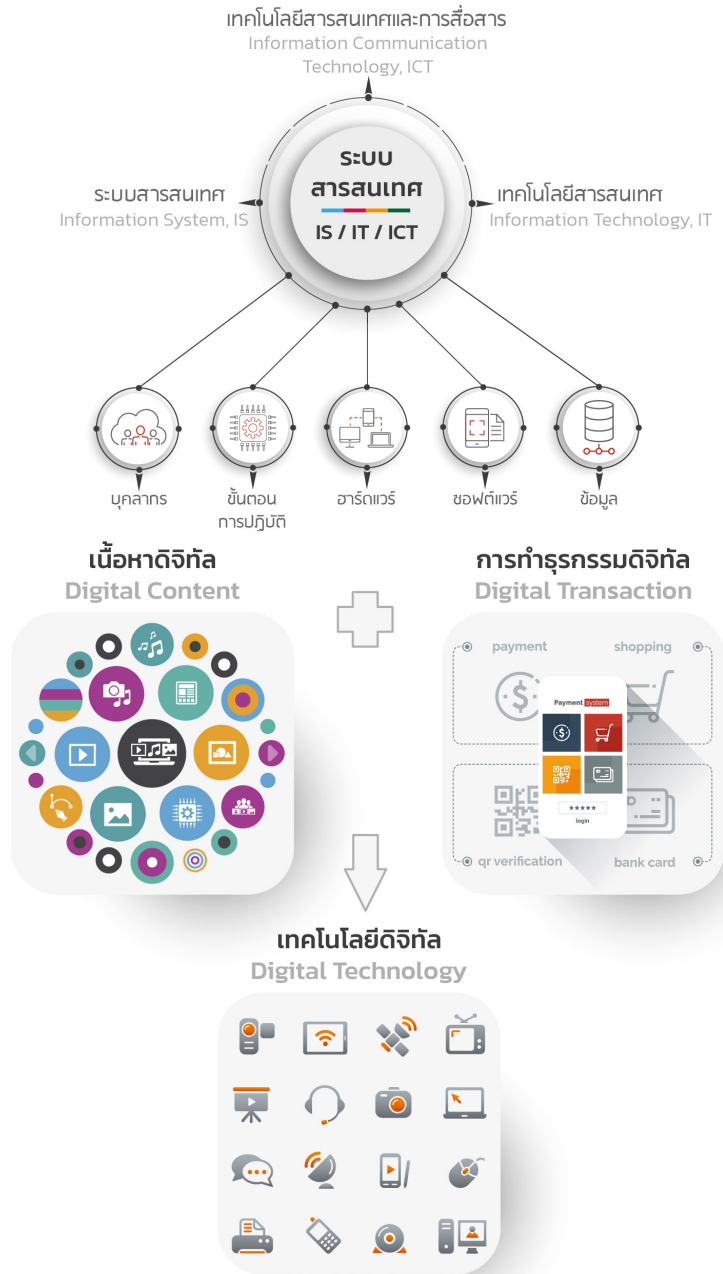
การปรับเปลี่ยนสู่ดิจิทัล DIGITAL TRANSFORMATION

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายรายละเอียดเกี่ยวกับเศรษฐกิจดิจิทัล (Digital Economy) องค์กรดิจิทัล (Digital Enterprise)
- เพื่ออธิบายแนวคิดในการปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation)
- เพื่อแนะนำหลักการและความสำคัญของวิศวกรรมซอฟต์แวร์ (Software Engineering)
- เพื่ออธิบายการประยุกต์หลักการของวิศวกรรมซอฟต์แวร์กับการพัฒนาโปรแกรมประยุกต์ เพื่อนำไปสู่การปรับเปลี่ยนสู่ดิจิทัล

เนื้อหาประจำบท (Contents)

- 1.1 เศรษฐกิจดิจิทัล (Digital Economy)
- 1.2 ความหมายของเทคโนโลยีดิจิทัล (Digital Technology Definition)
- 1.3 องค์กรดิจิทัล (Digital Enterprises)
- 1.4 การปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation)
- 1.5 บทบาทของวิศวกรรมซอฟต์แวร์กับการปรับเปลี่ยนสู่ดิจิทัล
- 1.6 คำนิยามและคำจำกัดความที่ใช้ในวิศวกรรมซอฟต์แวร์



รูปที่ 1.1 องค์ประกอบของเทคโนโลยีดิจิทัล



1.4 การปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation)

การปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation) คือ การปรับเปลี่ยนองค์กรเพื่อให้เข้ากับเศรษฐกิจดิจิทัล (Digital Economy) ด้วยเทคโนโลยีดิจิทัล (Digital Technology) ที่เป็นการปรับเปลี่ยนองค์กรให้ใช้ประโยชน์จากเทคโนโลยีดิจิทัล (Digitalization) ตั้งแต่การกำหนดเป้าหมาย วางแผน พัฒนาผลิตภัณฑ์หรือบริการ ตลอดจนช่องทางการส่งมอบสินค้าและบริการให้กับผู้บริโภค เพื่อเปลี่ยนองค์กรให้เป็นองค์กรดิจิทัลที่มีกระบวนการความคิดแบบดิจิทัล (Digital Mindset) และพัฒนาองค์กรให้เติบโตด้วยเทคโนโลยีดิจิทัล



รูปที่ 1.2 ขั้นตอนในการปรับเปลี่ยนสู่ดิจิทัล [2]

1.4.1 ขั้นตอนการปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation Roadmap)

แผนงานสู่การปรับเปลี่ยนสู่ดิจิทัล Earley Information Science, Inc. สรุปกระบวนการไว้ 4 ขั้นตอน [1] ดังแสดงในรูปที่ 1.3 คือ

STEP 1 Current State Assessment

สำรวจและประเมินสถานภาพปัจจุบันขององค์กรอย่างรอบด้าน

STEP 2 Future Vision

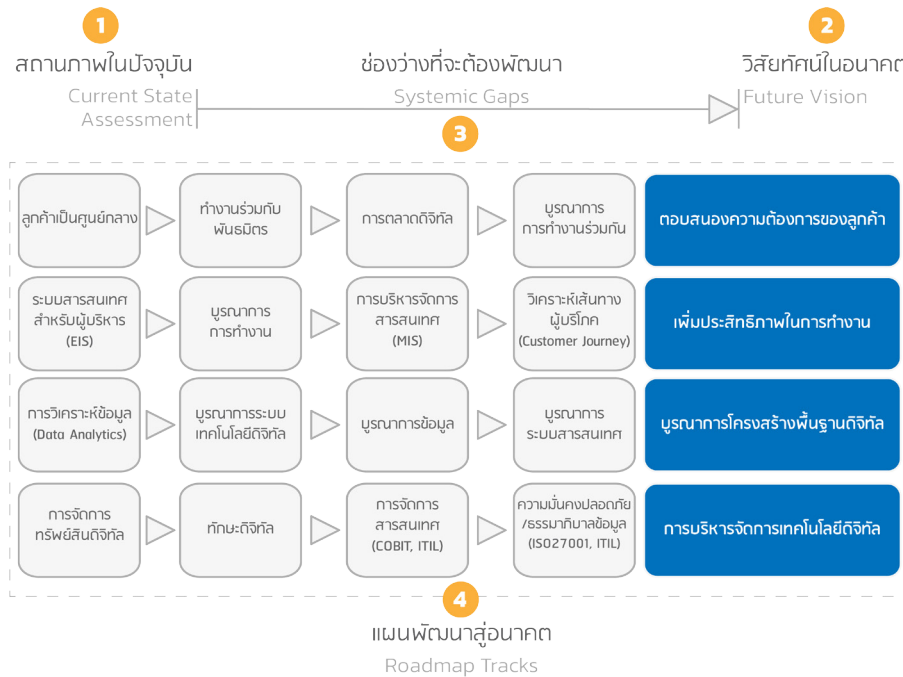
กำหนดวิสัยทัศน์ในอนาคตสำหรับการดำเนินงานแบบดิจิทัลอย่างชัดเจน

STEP 3 Systemic Gaps

วิเคราะห์ช่องว่างระหว่างสถานภาพในปัจจุบันกับวิสัยทัศน์ในอนาคต

STEP 4 Roadmap Tracks

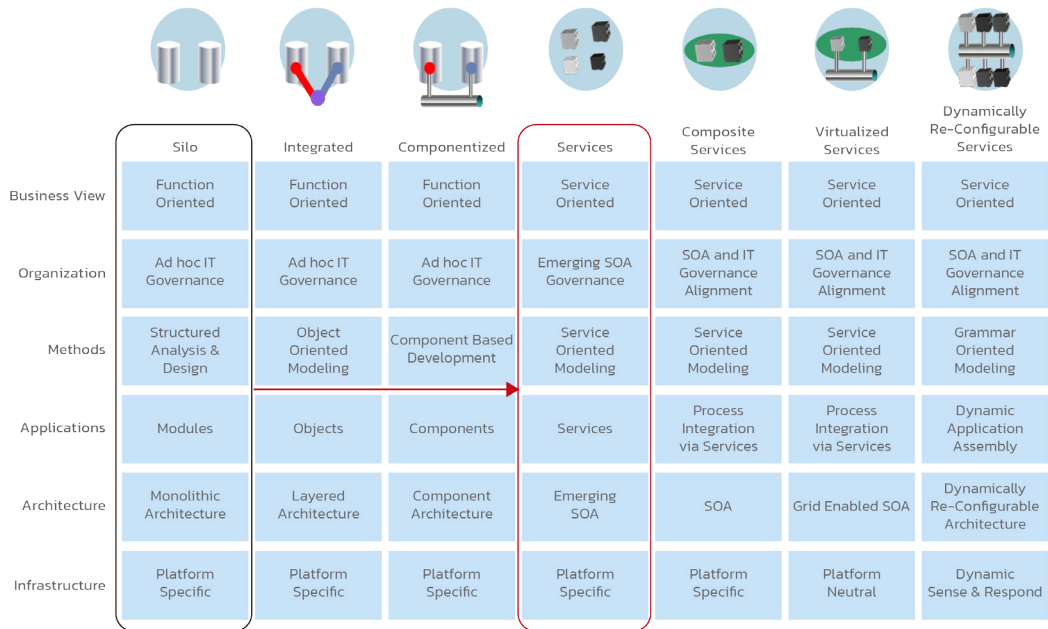
สร้างแนวทางหรือกลยุทธ์ตามปัจจัย 4 คือ 1) บุคลากร (People) 2) กระบวนการทำงาน (Process) 3) เทคโนโลยีดิจิทัล (Digital Technology) และ 4) การบริหารจัดการเทคโนโลยีดิจิทัล (Digital Technology Management) และธรรมาภิบาล (Governance)



รูปที่ 1.3 ช่องว่าง (Gap) ที่ต้องพัฒนาเพื่อการปรับเปลี่ยนสู่ดิจิทัล

ผู้บริหารในองค์กรเข้าใจผิดคิดว่า การปรับเปลี่ยนสู่ดิจิทัล (Digital Transformation) คือ เรื่องของเทคโนโลยี เป็นเรื่องของแผนกคอมพิวเตอร์ หรือเทคโนโลยีสารสนเทศ และคิดว่าการลงทุนด้านเทคโนโลยีดิจิทัลทำให้เกิดการพัฒนาไปสู่การเป็น องค์กรดิจิทัล (Digital Enterprise) ได้ ทั้งๆ ที่โดยแท้จริงแล้ว การปรับเปลี่ยนสู่ดิจิทัลคือเรื่องของ กลยุทธ์องค์กร สิ่งที่ต้องพัฒนาเพื่อให้บรรลุเป้าหมายที่ต้องการคือ “ดิจิทัลิเซชัน” (Digitization) และ “ดิจิทัลไลเซชัน” (Digitalization) การพัฒนาระบบด้วยหลักการทางวิศวกรรมซอฟต์แวร์เพียงอย่างเดียวจะเป็นเพียงการแปลงข้อมูลต่างๆ ให้อยู่ในรูปดิจิทัล อาจเป็นเพียงขั้นตอนของการทำดิจิทัลิเซชัน ดังนั้น ถ้าจะทำดิจิทัลไลเซชันซึ่งคือ การนำเทคโนโลยีดิจิทัลเข้ามา แล้วทำให้กระบวนการทำงานเปลี่ยนไป มีการทำงานที่มีประสิทธิภาพขึ้น ข้อสำคัญคือ “การเปลี่ยนประสบการณ์ของลูกค้า” (Customer Experience) ผ่าน “การเดินทางของลูกค้า” (Customer Journey) ซึ่งการปรับเปลี่ยนสู่ดิจิทัลก็คือผลลัพธ์ที่ได้จากการทำดิจิทัลไลเซชันนั่นเอง

ตัวอย่างเช่น การให้บริการของหน่วยงานส่วนใหญ่มุ่งเน้นการพัฒนาบริการทางอิเล็กทรอนิกส์ (E-Services) โดยการวิเคราะห์ และพัฒนาจากกระบวนการทำงานแบบเดิมๆ บริบทเดิมๆ ซึ่งทำให้ได้ซอฟต์แวร์ที่ทำงานเหมือนเดิม เพียงเปลี่ยนจากการกรอกในแบบฟอร์มที่เป็นกระดาษมาเป็นกรอกผ่านเว็บเท่านั้น ส่วนการทำงานอื่นยังเหมือนเดิม เช่น ต้องเดินทางไปที่หน่วยงานเพื่อชำระเงิน หรือต้องไปธนาคารเพื่อโอนเงิน เป็นต้น หน่วยงานยังคงต้องการเอกสารมากมาย เช่น สำเนาบัตรประชาชน ทะเบียนบ้าน โดยให้ประชาชนอัปโหลดเอกสารเหล่านั้นมาทางอินเทอร์เน็ต เป็นต้น



รูปที่ 1.6 การปรับเปลี่ยนสู่ดิจิทัลในบริบทเทคโนโลยีสารสนเทศจากแบบโซลูแบบเซอร์วิส

4.3.3 มีโครงสร้างพื้นฐานด้านเทคโนโลยีดิจิทัลที่รองรับการทำงานตามบริบทของระบบงานแต่ละ

4.4 ระบบนิเวศ (Ecosystem) องค์กรที่ประสบความสำเร็จในการปรับเปลี่ยนสู่ดิจิทัลส่วนมากจะไม่ได้ทำงานตามลำพัง แต่อาจอาศัยพันธมิตรในระบบนิเวศดิจิทัล และมีการสร้างแพลตฟอร์มทางธุรกิจที่หลากหลาย ฝ่ายมีส่วนร่วมและได้รับประโยชน์ทุกฝ่าย

4.4.1 การจัดการทรัพย์สินดิจิทัล (Digital Asset Management) มีการใช้เครื่องมือในการบริหารทรัพย์สินทางปัญญา เช่น ข้อความ ไฟล์รูปภาพ เนื้อหาดิจิทัล เป็นต้น หรือไม่

4.4.2 การจัดการสารสนเทศผลิตภัณฑ์ (Product Information Management) มีการจัดการข้อมูลผลิตภัณฑ์ไว้เป็นระบบ เพื่อนำเสนอลูกค้าได้อย่างเหมาะสมหรือไม่

4.4.3 การเพิ่มประสิทธิภาพการค้นหาข้อมูลในองค์กร มีการใช้เครื่องมือค้นหาที่ดีเพื่อให้สามารถเข้าถึงสารสนเทศต่างๆ ภายในหน่วยงานได้อย่างรวดเร็ว และระบบจัดการความรู้ (Knowledge Management) ของทั้งองค์กรหรือไม่

ทั้งนี้ แนวทางการปรับเปลี่ยนสู่ดิจิทัล อาจเริ่มต้นในลักษณะการถ่ายทอดกลยุทธ์จากผู้บริหารสูงสุด (Top-Down) หรืออาจเริ่มจากหน่วยธุรกิจใดธุรกิจหนึ่ง (Bottom-Up) หรือบางครั้งก็อาจเริ่มต้นด้วยการวิเคราะห์การเดินทางของลูกค้า (Customer Journey) แล้วมาพิจารณาว่าเทคโนโลยีดิจิทัลจะช่วยทำให้เกิดการเปลี่ยนแปลงได้อย่างไร

CHAPTER

2

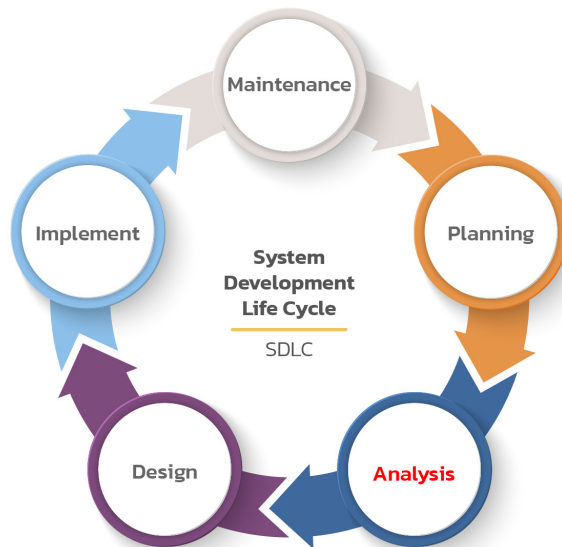
การวิเคราะห์เชิงโครงสร้าง STRUCTURED ANALYSIS

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายแนวคิดเชิงโครงสร้าง
- เพื่ออธิบายการวิเคราะห์เชิงโครงสร้าง
- เพื่ออธิบายแผนภาพการไหลข้อมูล (Data Flow Diagram, DFD)

เนื้อหาประจำบท (Content)

- 2.1 การวิเคราะห์เชิงโครงสร้าง (Structure Analysis and Design)
- 2.2 แผนภาพการไหลของข้อมูล (Data Flow Diagram)
- 2.3 แผนภาพการไหลของข้อมูลกับการปรับโครงสร้างทางธุรกิจ



System Development Life Cycle หรือ SDLC Process

2.1 การวิเคราะห์เชิงโครงสร้าง (Structure Analysis And Design)

การวิเคราะห์เชิงโครงสร้างได้รับการพัฒนาและเริ่มใช้ในชวงปี ค.ศ. 1969-1973 โดย Douglas T. Ross และ SofTech, Inc. [7] วิธีการนี้เป็นส่วนหนึ่งของชุดวิธีการที่มีโครงสร้าง ซึ่งแสดงถึงการวิเคราะห์การออกแบบ และเทคนิคการเขียนโปรแกรมที่ได้รับการพัฒนาขึ้นเพื่อตอบสนองต่อปัญหาที่โลกซอฟต์แวร์เผชิญตั้งแต่ปี ค.ศ. 1960-1980 โดยในเวลานั้นการเขียนโปรแกรมเชิงพาณิชย์ส่วนใหญ่ทำในภาษาโคบอล (COBOL) ฟอ์แทรน (FORTRAN) ตามด้วยภาษาปาสคาล (Pascal) ภาษาเบสิก (BASIC) และภาษาซี (C) แต่เนื่องจากวิธีการนี้ไม่มีเทคนิคมาตรฐานในการจัดทำในการวิเคราะห์และการออกแบบระบบ ทำให้การพัฒนาระบบสารสนเทศทำได้ยากขึ้นเรื่อยๆ เมื่อระบบมีขนาดใหญ่ขึ้นและซับซ้อนมากขึ้น อย่างไรก็ตาม วิธีการนี้ยังมีใช้อยู่ในปัจจุบันในกรณีที่พัฒนาระบบด้วยภาษาเชิงโครงสร้าง เช่น การใช้ภาษาซีในการพัฒนาระบบอินเทอร์เน็ตประสานสรรพสิ่ง (Internet of Things, IoT) เป็นต้น

การวิเคราะห์เชิงโครงสร้าง (Structure Analysis and Design) เป็นการวิเคราะห์การทำงานตามหน้าที่ (Function) ของแต่ละส่วนงานหรือแต่ละระบบย่อย เพื่ออธิบายการนำเข้าข้อมูล กระบวนการทำงาน และการแสดงผล รวมทั้งปฏิสัมพันธ์กับผู้ที่เกี่ยวข้อง เครื่องมือที่ช่วยในการวิเคราะห์ระบบด้วยวิธีนี้คือ แผนภาพการไหลของข้อมูล (Data Flow Diagram)

CHAPTER

3

การวิเคราะห์และออกแบบเชิงวัตถุ OBJECT-ORIENTED ANALYSIS AND DESIGN

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายกรอบความคิดเชิงวัตถุ (Object-Oriented Approach)
- เพื่ออธิบายการวิเคราะห์ระบบเชิงวัตถุ (Object-Oriented Analysis)
- เพื่ออธิบายการออกแบบเชิงวัตถุ (Object-Oriented Design)

เนื้อหาประจำบท (Contents)

- 3.1 กระบวนทัศน์เชิงวัตถุ (Object-Oriented Paradigm)
- 3.2 หลักการเชิงวัตถุ (Object-Oriented Concept)
- 3.3 นามธรรม (Abstraction)
- 3.4 การห่อหุ้ม (Encapsulation)
- 3.5 ความเป็นโมดูลาร์ (Modularity)
- 3.6 ลำดับชั้น (Hierarchy)
- 3.7 โพลิมอร์ฟิซึม (Polymorphism)
- 3.8 การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design)

ตารางที่ 3.1 การเปรียบเทียบกระบวนการค้นเชิงโครงสร้างกับวิธีการเชิงวัตถุ

	กระบวนการค้นเชิงโครงสร้าง	กระบวนการค้นเชิงวัตถุ
ลักษณะทั่วไป	แตกปัญหาออกเป็นส่วนย่อยในรูปของกระบวนการทำงาน	พิจารณาสิ่งต่างๆ เป็นวัตถุ ซึ่งมีความเป็นอิสระต่อกัน แต่ทำงานร่วมกัน
ลักษณะการจำแนกงาน	แตกกระบวนการทำงานเป็นหน่วยย่อยๆ เรียกว่า ฟังก์ชัน	จำแนกวัตถุแล้วแบ่งกลุ่มของวัตถุตามคุณสมบัติของแต่ละวัตถุ
ความสัมพันธ์	ฟังก์ชันการทำงานต่างๆ จะมีลักษณะการทำงานขึ้นตรงต่อกัน มีการส่งพารามิเตอร์จากฟังก์ชันหนึ่งไปอีกฟังก์ชันหนึ่ง	วัตถุแต่ละอันมีความเป็นอิสระต่อกัน และติดต่อกันโดยการส่งข้อความ (Message) ถึงกัน
ขั้นตอนการทำงาน	การกำหนดขั้นตอนการทำงานในแต่ละฟังก์ชัน และความสัมพันธ์ระหว่างฟังก์ชัน	การกำหนดคุณสมบัติและพฤติกรรมให้วัตถุต่างๆ จากนั้นสร้างความสัมพันธ์ระหว่างวัตถุ

ตารางที่ 3.2 เปรียบเทียบขั้นตอนของวิธีการเชิงโครงสร้างกับวิธีการเชิงวัตถุ

วิธีการเชิงโครงสร้าง	วิธีการเชิงวัตถุ
<ol style="list-style-type: none"> 1. การวิเคราะห์ความต้องการ (Requirement) 2. การวิเคราะห์ระบบ (Analysis) <ul style="list-style-type: none"> ● กำหนดสิ่งที่จะพัฒนา ● การสร้างแบบจำลองโดยใช้แผนภาพการไหลของข้อมูล (Data Flow Diagram) 3. การออกแบบระบบ (Design) <ul style="list-style-type: none"> ● ออกแบบสถาปัตยกรรม ● ออกแบบฐานข้อมูลและส่วนติดต่อผู้ใช้ 4. การทำให้เกิดผล (Implementation) <ul style="list-style-type: none"> ● พัฒนาโปรแกรมเชิงโครงสร้าง (C, COBOL) ● การทดสอบและติดตั้งระบบ 5. การบำรุงรักษา 	<ol style="list-style-type: none"> 1. การวิเคราะห์ความต้องการ (Requirement) 2. การวิเคราะห์ระบบเชิงวัตถุ <ul style="list-style-type: none"> ● การสร้างแบบจำลองโดยใช้ยูเอ็มแอล ● กำหนดยูสเคส (Use Case) คลาส (Class) ออบเจกต์ (Object) 3. การออกแบบระบบเชิงวัตถุ <ul style="list-style-type: none"> ● ออกแบบสถาปัตยกรรม ● ออกแบบฐานข้อมูลและส่วนติดต่อผู้ใช้ 4. การทำให้เกิดผลเชิงวัตถุ <ul style="list-style-type: none"> ● พัฒนาโปรแกรมเชิงวัตถุ (Java, C++) ● การทดสอบและติดตั้งระบบ 5. การบำรุงรักษา

หมายเหตุ ยูเอ็มแอล (UML) คือ คำย่อของ Unified Modeling Language รายละเอียดอยู่ในบทที่ 4

CHAPTER

4

UML ภาษาการสร้างแบบจำลอง แบบครบวงจร

วัตถุประสงค์ (Objectives)

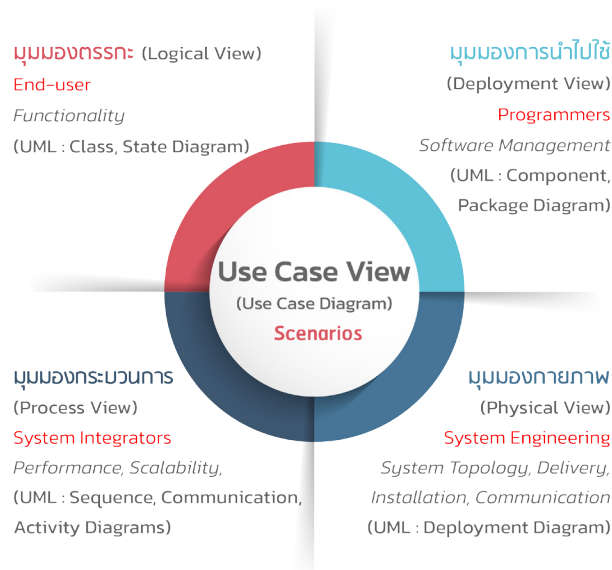
- เพื่ออธิบายภาษาการสร้างแบบจำลอง
- เพื่ออธิบายรายละเอียดของภาษาการสร้างแบบจำลองแบบครบวงจรแบบต่างๆ
- เพื่ออธิบายวิธีการทำแผนภาพแบบต่างๆ
- เพื่ออธิบายวิธีการพัฒนาระบบด้วยภาษาการสร้างแบบจำลอง

เนื้อหาประจำบท (Contents)

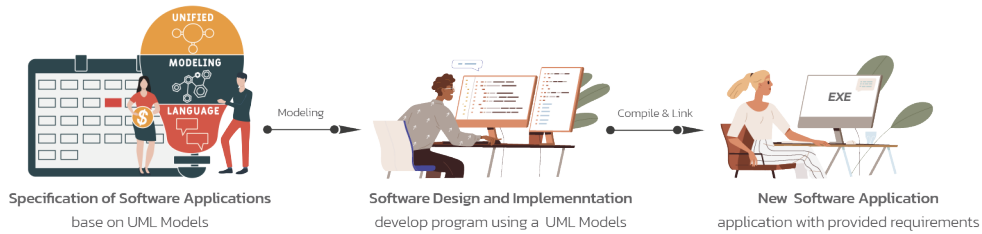
- 4.1 ภาษาการสร้างแบบจำลอง (Modeling Language)
- 4.2 ความเป็นมาของภาษาการสร้างแบบจำลองแบบครบวงจร (Unified Modeling Language, UML)
- 4.3 แผนภาพยูเอ็มแอล (UML Diagram)
- 4.4 แผนภาพยูสเคส (Use Case diagram)
- 4.5 แผนภาพคลาส (Class Diagram)
- 4.6 แผนภาพลำดับ (Sequence Diagram)
- 4.7 แผนภาพสถานะ (State Diagram)
- 4.8 แผนภาพกิจกรรม (Activity Diagram)
- 4.9 แผนภาพการนำไปใช้ (Deployment Diagram)
- 4.10 การสร้างรายงานโดยยูเอ็มแอล
- 4.11 การพัฒนาโปรแกรมโดยใช้แบบจำลอง
- 4.12 การพัฒนาด้วยวิศวกรรมย้อนกลับ (Reverse Engineering)



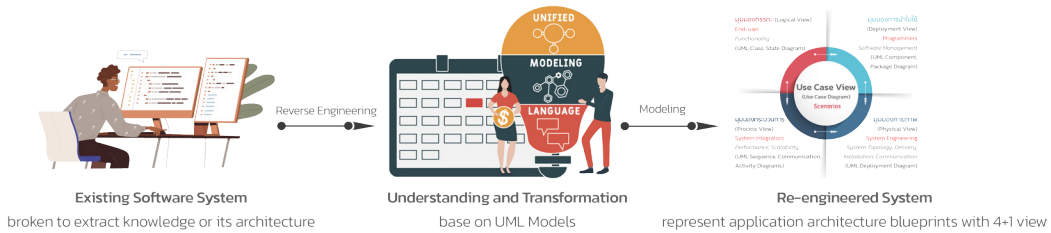
3. **มุมมองการนำไปใช้ (Deployment View)** มุมมองการพัฒนาแสดงให้เห็นถึงระบบจากมุมมองของโปรแกรมเมอร์ และผู้เกี่ยวข้องกับการจัดการซอฟต์แวร์ แผนภาพยูเอ็มแอลที่เกี่ยวข้องคือ แผนภาพส่วนประกอบ (Component Diagram) เพื่ออธิบายส่วนประกอบของระบบ และแผนภาพแพ็คเกจ (Package Diagram)
4. **มุมมองกายภาพ (Physical View)** แสดงให้เห็นถึงระบบจากมุมมองของวิศวกรระบบ ซึ่งเกี่ยวข้องกับทอพอโลยี (Topology) ของส่วนประกอบซอฟต์แวร์ที่เชื่อมต่อทางกายภาพระหว่างส่วนประกอบต่างๆ แผนภาพยูเอ็มแอลที่ใช้คือ แผนภาพการนำไปใช้ (Deployment Diagram)
5. **สถานการณ์จำลอง (Scenarios)** แสดงโดยใช้กรณีการใช้งาน หรือสถานการณ์จำลอง ซึ่งจะกลายเป็นมุมมองที่ให้อธิบายลำดับของการโต้ตอบระหว่างผู้กระทำ (Actor) และยูสเคส (Use Case) ใช้เพื่อระบุนอ้กประกอบทางสถาปัตยกรรม และตรวจสอบความถูกต้องของการออกแบบสถาปัตยกรรม นอกจากนี้ยังใช้เป็นจุดเริ่มต้นสำหรับการทดสอบต้นแบบสถาปัตยกรรม มุมมองนี้เรียกอีกอย่างว่า มุมมองยูสเคส โดยใช้แผนภาพยูสเคส (Use Case Diagram)



รูปที่ 4.4 แผนภาพยูเอ็มแอลที่สัมพันธ์กับแบบจำลองมุมมองสถาปัตยกรรมแบบ 4 + 1 [19]



รูปที่ 4.5 การพัฒนาแบบวิศวกรรมก้าวหน้า (Forward Engineering)



รูปที่ 4.6 การพัฒนาแบบวิศวกรรมย้อนกลับ (Reverse Engineering)

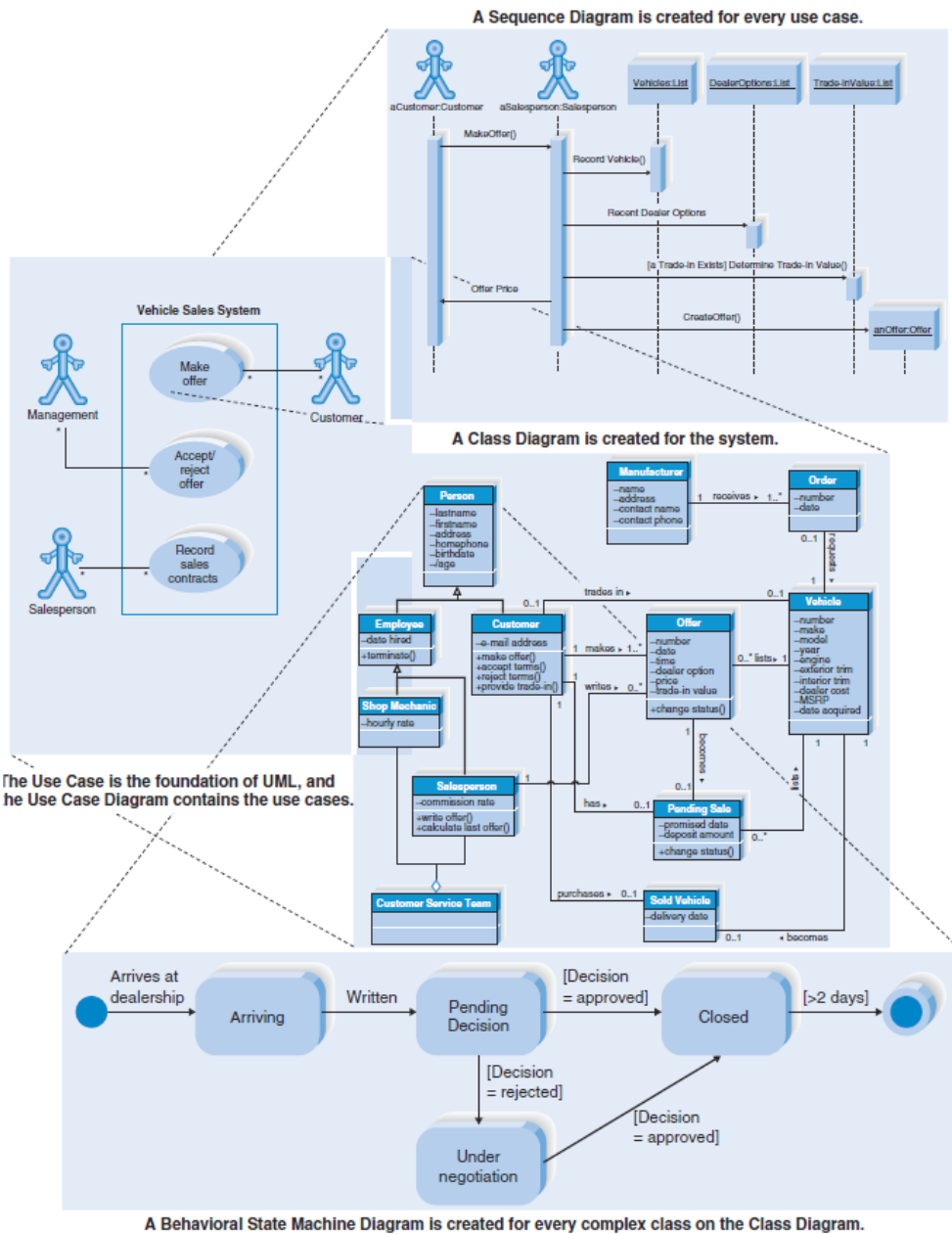
ข้อดีของการพัฒนาระบบด้วยภาษายูเอ็มแอล

- ยูเอ็มแอลเป็นภาษาการร่างแบบจำลองที่พร้อมใช้งานให้กับผู้ใช้เพื่อให้สามารถพัฒนาและแลกเปลี่ยนแบบจำลองที่มีความหมายได้
- ยูเอ็มแอลได้จัดเตรียมเครื่องมือที่ช่วยให้ขยายแนวคิด (Concepts) ได้อย่างไม่จำกัด
- ยูเอ็มแอลเป็นอิสระจากภาษาโปรแกรมและกระบวนการพัฒนา
- ยูเอ็มแอลให้พื้นฐานที่เป็นมาตรฐานสำหรับการทำความเข้าใจภาษาการร่างแบบจำลอง
- ส่งเสริมการเติบโตของตลาดเครื่องมือที่สนับสนุนแนวคิดเชิงวัตถุ
- สนับสนุนแนวคิดการพัฒนาที่สูงขึ้น เช่น การพัฒนาแบบร่วมมือ (Collaborations) การใช้เฟรมเวิร์ค (Frameworks) และคอมโพเนนต์ (Components)
- บูรณาการแนวทางปฏิบัติที่ดีที่สุดในการพัฒนาระบบ (Best Practice)

องค์ประกอบของภาษายูเอ็มแอล

ภาษาทั่วไปจะประกอบด้วยคำศัพท์และไวยากรณ์ ภาษายูเอ็มแอลก็เช่นเดียวกัน ประกอบด้วยคำศัพท์ 3 ส่วน ดังนี้ (รูปที่ 4.7)

1. **สัญลักษณ์ (Things)** เป็นรูปแบบที่เล็กที่สุดของแบบจำลอง
2. **ความสัมพันธ์ (Relationships)** เป็นสิ่งที่ใช้แสดงความสัมพันธ์ระหว่าง Things
3. **แผนภาพ (Diagrams)** ใช้จัดกลุ่มให้แก่ Things ที่สามารถจัดให้อยู่ในกลุ่มเดียวกันได้



รูปที่ 4.27 ความเชื่อมโยงของแผนภาพลำดับ (Sequence) กับแผนภาพอื่นๆ [5]

แผนภาพลำดับเป็นแผนภาพที่แสดงให้เห็นถึงการปฏิสัมพันธ์ระหว่างออบเจกต์ (Object) ณ เวลาต่างๆ ประกอบด้วยสัญลักษณ์ ดังนี้

CHAPTER

5

กระบวนการซอฟต์แวร์ SOFTWARE PROCESS



วัตถุประสงค์ (Objectives)

- เพื่อนำเสนอแนวคิดเกี่ยวกับกระบวนการซอฟต์แวร์
- เพื่อนำเสนอรายละเอียดเกี่ยวกับรูปแบบการซอฟต์แวร์แบบต่างๆ
- เพื่ออธิบายกิจกรรมพื้นฐานที่จำเป็นในการพัฒนาซอฟต์แวร์
- เพื่ออธิบายปัจจัยที่มีผลต่อคุณภาพของซอฟต์แวร์
- เพื่ออธิบายการปรับปรุงประสิทธิภาพของซอฟต์แวร์

เนื้อหาประจำบท (Contents)

- 5.1 แบบจำลองกระบวนการซอฟต์แวร์ (Software Process Models)
- 5.2 กิจกรรมในกระบวนการซอฟต์แวร์ (Process Activities)



5.1 แบบจำลองกระบวนการซอฟต์แวร์ (Software Process Models)

แบบจำลองกระบวนการซอฟต์แวร์ (Software Process Models) เป็นการแสดงกระบวนการซอฟต์แวร์ให้มีความชัดเจนขึ้น โดยแสดงถึงกระบวนการทั้งหมดที่จะเกิดขึ้นในการพัฒนาซอฟต์แวร์ รวมถึงกิจกรรม กระบวนการ และลำดับของกิจกรรม แต่อาจไม่แสดงบทบาทของคนที่เกี่ยวข้องในกิจกรรมเหล่านี้

อย่างไรก็ตาม แบบจำลองกระบวนการซอฟต์แวร์อาจไม่แสดงรายละเอียดของกิจกรรมเฉพาะที่ชัดเจนของกระบวนการซอฟต์แวร์ โดยเฉพาะส่วนที่เป็นนามธรรมซึ่งต้องอาศัยกระบวนการอื่นเพิ่มเติมที่สามารถใช้อธิบายด้วยวิธีการที่แตกต่างกันไป

แบบจำลองกระบวนการซอฟต์แวร์ ที่จะกล่าวถึงในที่นี้ประกอบด้วย

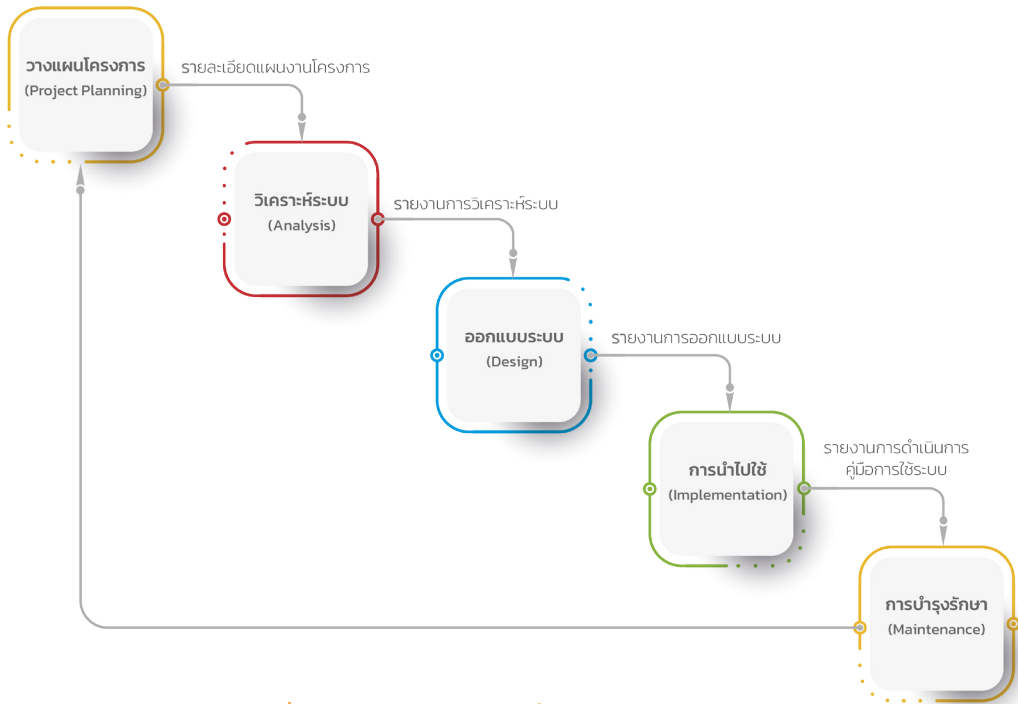
1. แบบจำลองการพัฒนาแบบน้ำตก (Water Fall Model)
2. แบบจำลองการพัฒนาต้นแบบ (Prototyping)
3. แบบจำลองการพัฒนาแบบมีส่วนร่วม (Joint Application Development, JAD)
4. แบบจำลองการพัฒนาแบบอย่างรวดเร็ว (Rapid Application Development, RAD)
5. การทำงานแบบอไจล์ (Agile Methodology)
6. แบบจำลองการพัฒนาแบบเอ็กซ์ตรีม (Extreme Programming)
7. แบบจำลองการพัฒนาโดยวิธีเชิงวัตถุ (Object Oriented Development)
8. สถาปัตยกรรมเชิงเซอร์วิส (Service Oriented Architecture)

5.1.1 แบบจำลองการพัฒนาแบบน้ำตก (Water Fall Model)

แบบจำลองการพัฒนาแบบน้ำตก (Water Fall Model) คือ กระบวนการทางความคิด (Logical Process) ในการพัฒนาระบบสารสนเทศเพื่อแก้ปัญหาทางธุรกิจ และตอบสนองความต้องการของผู้ใช้ได้ โดยระบบที่จะพัฒนานั้นอาจเริ่มด้วยการพัฒนาระบบใหม่เลย หรือนำระบบเดิมที่มีอยู่แล้วมาปรับเปลี่ยนให้ดียิ่งขึ้น ขั้นตอนในวงจรการพัฒนาแบบช่วยให้นักวิเคราะห์ระบบสามารถดำเนินการได้อย่างมีแนวทางและเป็นขั้นตอน ทำให้สามารถควบคุมระยะเวลา และงบประมาณในการปฏิบัติงานของโครงการพัฒนาระบบได้

ระบบสารสนเทศทั้งหลายมีวงจรชีวิตเหมือนกันตั้งแต่เกิดจนตาย วงจรนี้จะเป็นขั้นตอนที่เป็นลำดับตั้งแต่เริ่มต้นจนเสร็จสิ้นเป็นระบบที่ใช้งานได้ ซึ่งนักวิเคราะห์ระบบต้องทำความเข้าใจให้ดีว่า ในแต่ละขั้นตอนจะต้องทำอะไร และทำอย่างไร

ขั้นตอนการพัฒนาแบบน้ำตก (Water Fall Model) มีอยู่ด้วยกัน 5 ขั้นตอน ดังแสดงในรูปที่ 5.1 ดังนี้



รูปที่ 5.1 วงจรการพัฒนาแบบน้ำตก (Water Fall Model) [2, 19]

STEP 1 วางแผนโครงการ (Project Planning)

เป็นการวางแผนการพัฒนาสารสนเทศ โดยการระบุปัญหา โอกาส และจุดมุ่งหมาย ซึ่งเป็นขั้นตอนที่สำคัญเป็นการกำหนดทิศทางในการพัฒนาให้ชัดเจน เช่น การระบุปัญหาจะได้มาจากพนักงานที่ทำงานแล้วพบว่างานที่ทำมีปัญหาเกิดขึ้น หรือไม่พอใจกับระบบการทำงานเดิม ในการระบุโอกาสคือการสังเกตว่าลักษณะงานเดิมสามารถนำระบบสารสนเทศมาปรับปรุงให้ทำงานสะดวกรวดเร็วได้หรือไม่ สามารถเพิ่มประสิทธิภาพประสิทธิผลในการทำงาน หรือสู้กับคู่แข่งในด้านสารสนเทศได้อย่างไร และการระบุจุดมุ่งหมายโดยดูจุดมุ่งหมายหลักขององค์กร เช่น การลดต้นทุน การลดจำนวนสินค้าที่จัดเก็บ เป็นต้น

CHAPTER

6

การจัดการเทคโนโลยีดิจิทัล DIGITAL TECHNOLOGY MANAGEMENT

วัตถุประสงค์ (Objectives)

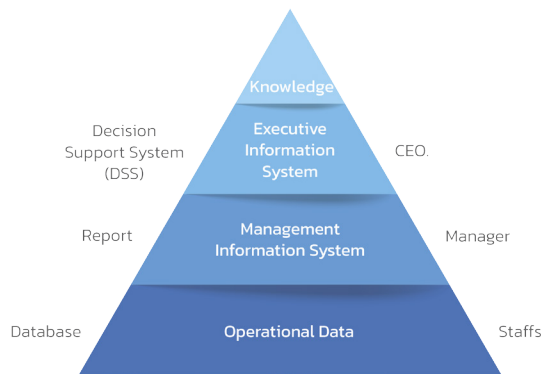
- เพื่ออธิบายปัญหาในการบริหารจัดการองค์กรและเทคโนโลยีดิจิทัล
- เพื่ออธิบายรายละเอียดสถาปัตยกรรมองค์กรในการปรับเปลี่ยนสู่ดิจิทัล
- เพื่ออธิบายความเชื่อมโยงของสถาปัตยกรรมองค์กรกับวิศวกรรมซอฟต์แวร์
- เพื่ออธิบายการบริหารจัดการเทคโนโลยีดิจิทัลให้มีประสิทธิภาพ
- เพื่ออธิบายการจัดการโครงการด้านซอฟต์แวร์
- เพื่ออธิบายความเสี่ยงและการจัดการความเสี่ยง
- เพื่ออธิบายการจัดการคุณภาพของซอฟต์แวร์

เนื้อหาประจำบท (Contents)

- 6.1 ปัญหาในการบริหารจัดการองค์กร
- 6.2 สถาปัตยกรรมองค์กร (Enterprise Architecture)
- 6.3 สถาปัตยกรรมองค์กรกับวิศวกรรมซอฟต์แวร์ (Enterprise Architecture and Software Engineering)
- 6.4 การกำกับดูแลการใช้เทคโนโลยีดิจิทัล
- 6.5 การจัดการซอฟต์แวร์ (Software Management)
- 6.6 การปรับปรุงคุณภาพซอฟต์แวร์ให้มีประสิทธิภาพ
- 6.7 ปัจจัยสู่ความสำเร็จ (Keys Success Factors)

6.1 ปัญหาในการบริหารจัดการองค์กร

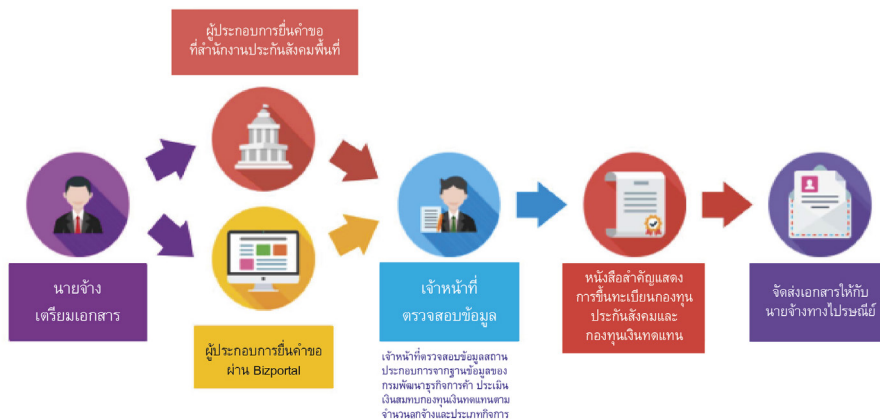
ประเทศไทยที่มีการบริหารจัดการแบบบนลงล่าง (Top-Down Management) และมักจะแก้ไขปัญหาระยะสั้นมากกว่าปัญหาระยะปานกลางหรือระยะยาว ดังนั้น ความต้องการข้อมูลหรือสารสนเทศจึงต้องเน้นการตอบปัญหาของผู้บริหารระดับสูง (CEO) ในลักษณะสารสนเทศเพื่อผู้บริหารระดับสูง (Executive Information System, EIS) หรือสนับสนุนการตัดสินใจ (Decision Support System, DSS) สำหรับผู้บริหารระดับกลางจะใช้สารสนเทศเพื่อการจัดการ (Management Information System, MIS) ในขณะที่เดียวกันก็ต้องรองรับการทำงานของระดับปฏิบัติการด้วยระบบฐานข้อมูล (Database System) ดังแสดงในรูปที่ 6.1



รูปที่ 6.1 ระดับชั้นของการพัฒนาสารสนเทศ

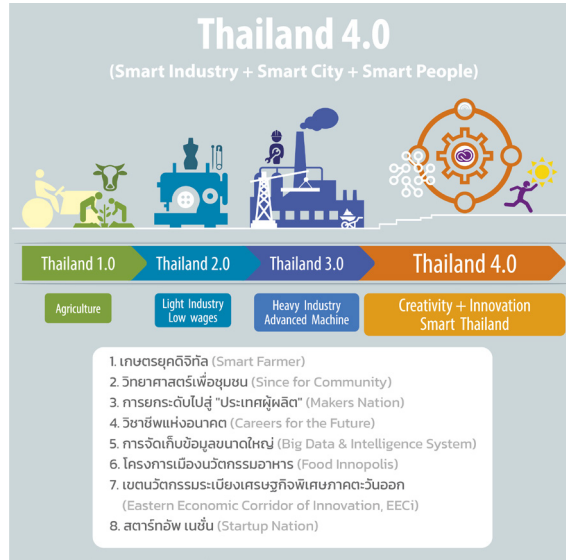
ปัญหาในการบริหารจัดการหน่วยงานหรือองค์กรต่างๆ มี 3 ลักษณะ กล่าวคือ

1. **ปัญหาตามภาระหน้าที่ หรือฟังก์ชันการทำงาน (Function Based)** คือ ปัญหาที่เกี่ยวข้องกับภารกิจหลักของหน่วยงาน ซึ่งต้องการระบบสารสนเทศที่รองรับระดับปฏิบัติการ (Operation) และระดับสารสนเทศเพื่อการจัดการ (MIS)



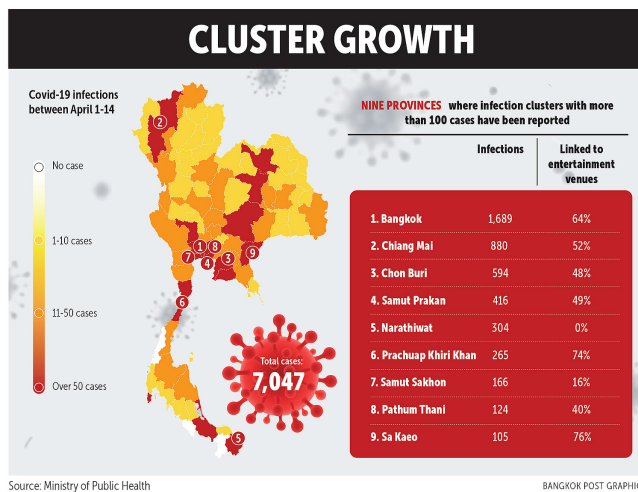
รูปที่ 6.2 ปัญหาเชิงฟังก์ชัน (Function Based Problems)

2. ปัญหาตามนโยบายหรือยุทธศาสตร์ (Agenda Based) เป็นปัญหาที่เกี่ยวข้องกับนโยบาย หรือยุทธศาสตร์ที่ปรับเปลี่ยนไปตามสถานการณ์ ระบบสารสนเทศที่รองรับปัญหานี้คือ ระบบสารสนเทศสำหรับผู้บริหาร (EIS) โดยใช้ระบบคลังข้อมูลหรือเหมืองข้อมูล (Data Warehouse/Data Mining)



รูปที่ 6.3 ปัญหาเชิงนโยบาย (Agenda Based Problems)

3. ปัญหาเชิงพื้นที่ (Area Based) เป็นปัญหาที่เกี่ยวข้องกับพื้นที่ มักจะเกิดกับหน่วยงานที่มีสาขาหลายแห่ง หรือมีการดำเนินงานที่เกี่ยวข้องกับพื้นที่ ระบบสารสนเทศที่ใช้กับระบบนี้คือ ระบบสารสนเทศภูมิศาสตร์ (GIS)

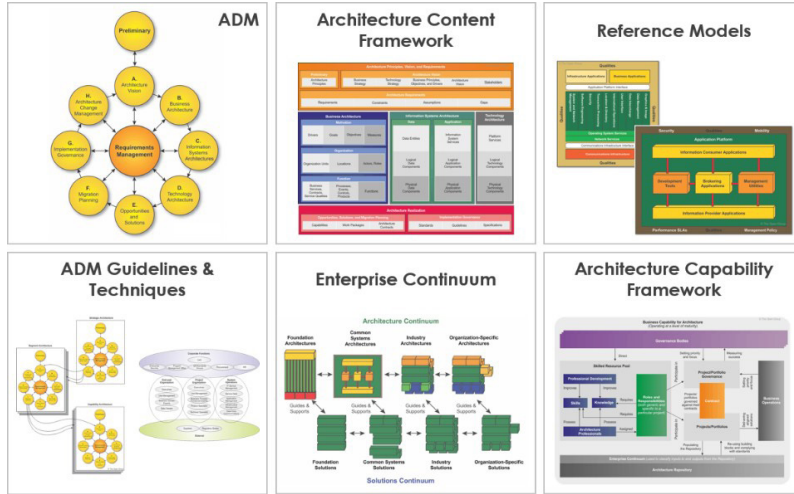


รูปที่ 6.4 ปัญหาเชิงพื้นที่ (Area Based Problems)

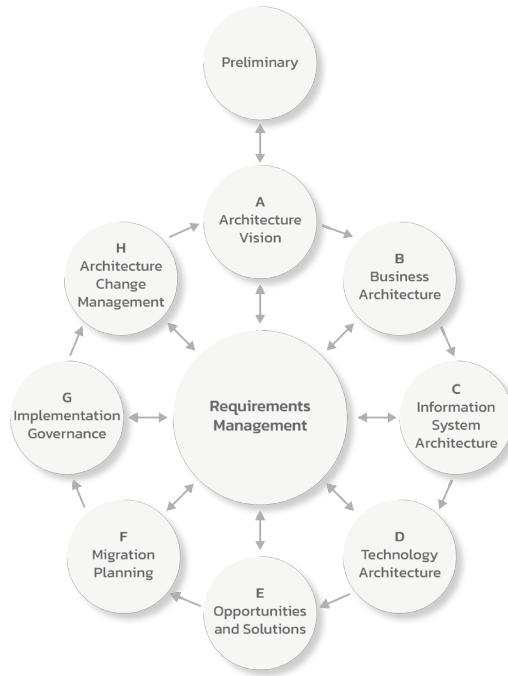
(ที่มา : www.bangkokpost.com)



6. **กรอบความสามารถทางสถาปัตยกรรม (Architecture Capability Framework)** ส่วนนี้จะกล่าวถึงกระบวนการทางทักษะ บทบาท และความรับผิดชอบที่จำเป็นในการสร้างและดำเนินการแนวปฏิบัติด้านสถาปัตยกรรมภายในองค์กร



รูปที่ 6.9 แสดงโครงสร้างและส่วนประกอบของ TOGAF [5]



รูปที่ 6.10 วิธีการพัฒนาสถาปัตยกรรม (Architecture Development Method) [5]

CHAPTER

7

วิศวกรรมความต้องการ REQUIREMENT ENGINEERING

วัตถุประสงค์ (Objectives)

- เพื่อให้เข้าใจแนวคิดของความต้องการของระบบและการเขียนข้อกำหนดของซอฟต์แวร์
- เพื่อให้เข้าใจความแตกต่างระหว่างข้อกำหนดความต้องการซอฟต์แวร์ทั้งแบบฟังก์ชัน (Functional) และแบบไม่เป็นฟังก์ชัน (Nonfunctional)
- เพื่อให้เข้าใจกิจกรรมหลักของวิศวกรรมความต้องการ ซึ่งประกอบด้วย การตอบสนองความต้องการ (Elicitation) การวิเคราะห์ และการตรวจสอบ รวมถึงความสัมพันธ์ระหว่างกิจกรรม
- เพื่อให้เข้าใจถึงวิธีการจัดทำเอกสารข้อกำหนดความต้องการด้านซอฟต์แวร์
- เพื่อให้เข้าใจถึงความจำเป็นในการจัดการความต้องการ
- เพื่อให้เข้าใจวิศวกรรมความต้องการทั้งแบบดั้งเดิมและแบบอไจล์

เนื้อหาประจำบท (Contents)

- 7.1 ความหมายของความต้องการด้านซอฟต์แวร์ (Software Requirement Definition)
- 7.2 กระบวนการวิศวกรรมความต้องการ (Requirement Engineering Process)
- 7.3 กระบวนการจัดทำข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirements Specification Process, SRS)
- 7.4 การสกัดและวิเคราะห์ความต้องการ (Requirements Elicitation and Analysis)
- 7.5 การตรวจสอบความต้องการ (Requirements Validation)
- 7.6 การจัดการความต้องการ (Requirements Management)
- 7.7 วิศวกรรมความต้องการแบบอไจล์ (Agile Requirement Engineering)

จากตารางที่ 7.1 แสดงให้เห็นว่า สาเหตุของปัญหาที่ทำให้โครงการซอฟต์แวร์ล้มเหลวเกี่ยวข้องกับการกำหนดความต้องการ (หัวข้อที่ 1, 2, 4, 6, 8) ถึง 51.6% ถึงแม้ในปัจจุบันมีการใช้รูปแบบที่สามารถรองรับการเปลี่ยนแปลงความต้องการที่เกิดขึ้น เช่น แบบอจิล (Agile) เป็นต้น แต่วิศวกรรมความต้องการยังมีความจำเป็น เพื่อให้เกิดการพัฒนาซอฟต์แวร์ที่มีแบบแผนและเป็นระบบ

7.1 ความหมายของความต้องการด้านซอฟต์แวร์ (Software Requirement Definition)

ตามการนิยามของ IEEE-STD-1220-1998 (IEEE 1998) [23] ได้ให้ความหมายไว้ว่า

ความต้องการด้านซอฟต์แวร์ (Software Requirement) คือ ข้อความที่ระบุถึงผลผลิต กระบวนการทำงาน ฟังก์ชันการออกแบบคุณลักษณะ รวมถึงข้อจำกัดต่างๆ ซึ่งมีการระบุเป็นที่ชัดเจน สามารถตรวจสอบได้หรือสามารถวัดได้เป็นรูปธรรม กระบวนการนี้จำเป็นสำหรับการตรวจรับซอฟต์แวร์โดยกระบวนการที่ยอมรับได้ และสามารถตรวจสอบคุณภาพของซอฟต์แวร์ได้ ความต้องการด้านซอฟต์แวร์โดยทั่วไปจะอยู่ในรูปแบบข้อกำหนดของซอฟต์แวร์ (Software Specification) ซึ่งอธิบายว่าระบบที่กำลังจะพัฒนามีการให้บริการ (Services) และมีข้อจำกัดอย่างไรในการดำเนินงาน ข้อกำหนดเหล่านี้จะสะท้อนความต้องการของผู้ใช้ กระบวนการในการค้นหาวิเคราะห์ จัดทำเอกสารข้อกำหนดและการตรวจสอบ รวมถึงข้อจำกัดต่างๆ เรียกว่า “วิศวกรรมความต้องการ” (Requirement Engineering)

ผลลัพธ์ของกระบวนการวิศวกรรมความต้องการนั้น สามารถแบ่งได้เป็นสองระดับคือ 1) ความต้องการของผู้ใช้ (User Requirements) ซึ่งหมายถึง ข้อกำหนดในระดับผู้ใช้งาน และ 2) ความต้องการของระบบ (System Requirements) หมายถึง คำอธิบายโดยละเอียดเกี่ยวกับสิ่งที่ระบบควรทำ มีรายละเอียด ดังนี้

1. ความต้องการของผู้ใช้ (User Requirements)

ข้อความที่พรรณนา รวมถึงแผนภาพของบริการที่ระบบคาดว่าจะให้กับผู้ใช้ระบบ และข้อจำกัดภายใต้การดำเนินการนั้น

2. ความต้องการของระบบ (System Requirements)

คำอธิบายโดยละเอียดเพิ่มเติมของระบบซอฟต์แวร์ฟังก์ชัน บริการ และข้อจำกัดในการดำเนินงาน เอกสารข้อกำหนดของระบบ บางครั้งเรียกว่า “ข้อกำหนดฟังก์ชัน” (Functional Specification)

การจัดทำข้อกำหนดนั้นควรกำหนดสิ่งที่แน่นอนที่จะดำเนินการ ซึ่งข้อกำหนดนี้เป็นส่วนหนึ่งของร่างขอบเขตงาน (Term of Reference, TOR) หรือสัญญาะหว่างผู้ใช้งานและผู้พัฒนาซอฟต์แวร์ ซึ่งผู้ที่มารับจ้างพัฒนาระบบจะได้รับทราบข้อมูลในทิศทางเดียวกัน เมื่อผู้รับจ้างพัฒนาได้ศึกษาร่างขอบเขตงาน (TOR) แล้ว ก็จะสามารถจัดทำข้อเสนอโครงการ (Proposal) ให้กับผู้ว่าจ้างได้อย่างถูกต้องโดยอ้างอิงจากข้อกำหนดซอฟต์แวร์

STEP 5 การตรวจสอบความต้องการ (Requirement Validation)

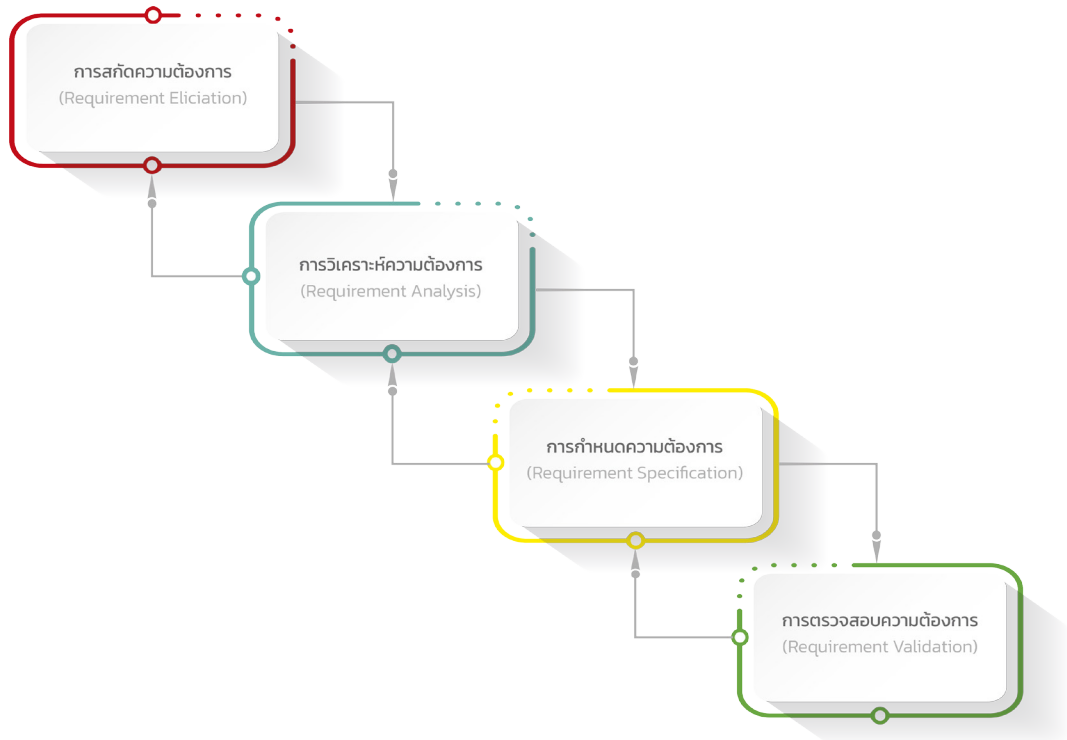
ในขั้นตอนนี้ ผู้ใช้จะพิจารณาว่า ข้อกำหนดความต้องการมีความเป็นไปได้ตามวัตถุประสงค์ที่วางไว้หรือไม่ ถ้ามีความเป็นไปได้ตามวัตถุประสงค์ที่วางไว้ จึงจะเริ่มดำเนินการขั้นตอนถัดไปคือ การออกแบบระบบ

STEP 6 การจัดการความต้องการ (Requirement Management)

ขั้นตอนนี้อาจจะไม่จำเป็นต้องมีขึ้นอยู่กับบริบทของแต่ละองค์กร ส่วนใหญ่จะเกิดกับองค์กรที่มีการพัฒนา หรือมีการเปลี่ยนแปลงความต้องการ

7.3 กระบวนการจัดทำข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirement Specification Process)

วิศวกรรมความต้องการ (Requirement Engineering) เป็นส่วนย่อยของวิศวกรรมซอฟต์แวร์โดยเน้นกระบวนการจัดทำข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirement Specification, SRS) ซึ่งเกิดจากการค้นหาและทำความเข้าใจในความต้องการของผู้ใช้ซึ่งเป็นนามธรรม ซึ่งประกอบด้วย 4 ขั้นตอน คือ



รูปที่ 7.3 กระบวนการจัดทำข้อกำหนดความต้องการด้านซอฟต์แวร์ (Software Requirement Specification Process) [22]

CHAPTER

8

การออกแบบระบบ SYSTEM DESIGN

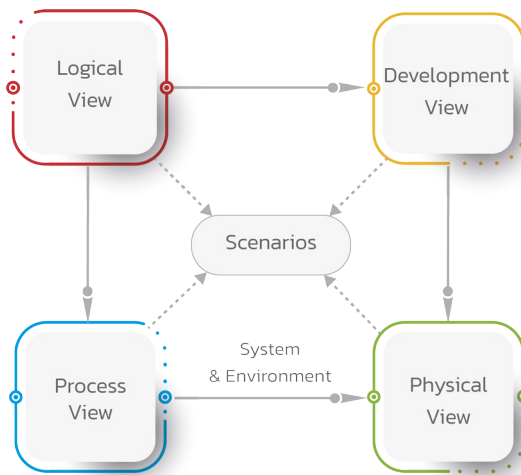
วัตถุประสงค์ (Objectives)

- เพื่ออธิบายกระบวนการออกแบบระบบ
- เพื่ออธิบายกระบวนการออกแบบสถาปัตยกรรม
- เพื่ออธิบายกระบวนการออกแบบส่วนเชื่อมต่อ
- เพื่ออธิบายกระบวนการออกแบบส่วนประกอบ
- เพื่ออธิบายกระบวนการออกแบบฐานข้อมูล
- เพื่ออธิบายรูปแบบการออกแบบ

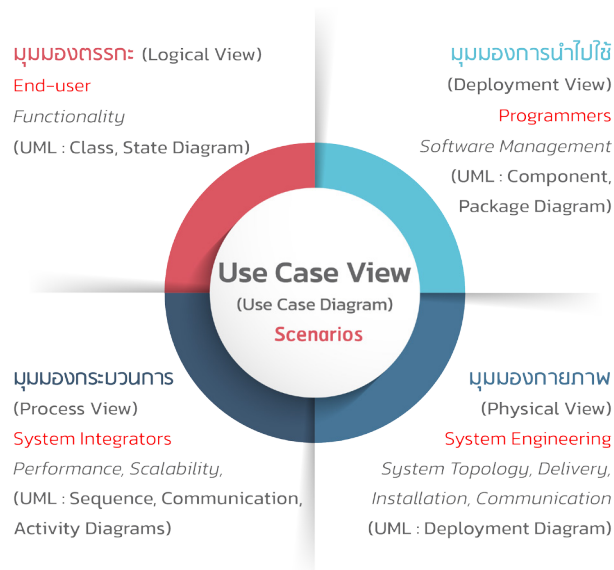
เนื้อหาประจำบท (Contents)

- 8.1 กระบวนการออกแบบระบบ (System Design)
- 8.2 การออกแบบสถาปัตยกรรม (Architectural Design)
- 8.3 การออกแบบส่วนต่อประสาน (Interface Design)
- 8.4 การออกแบบส่วนประกอบ (Component Design)
- 8.5 การออกแบบฐานข้อมูล (Database Design)
- 8.6 รูปแบบการออกแบบ (Design Pattern)

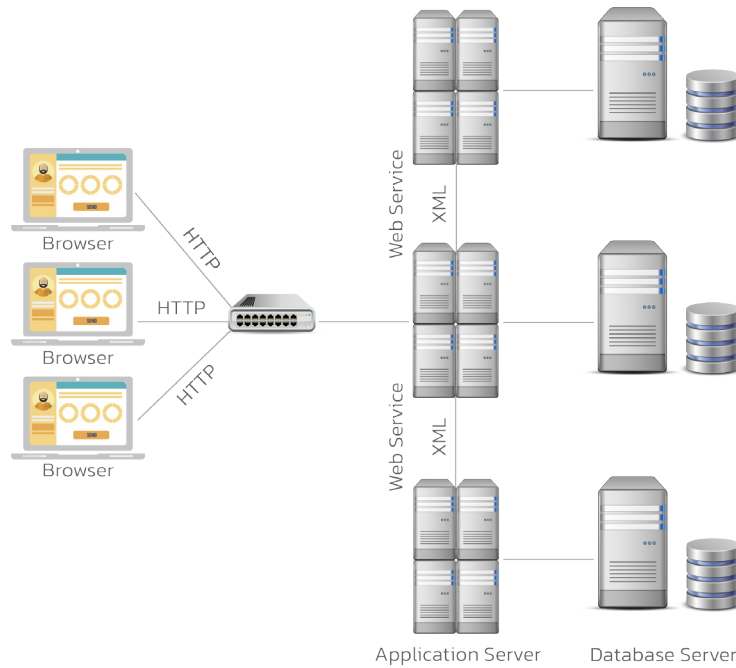
5. สถานการณ์จำลอง (Scenarios) แสดงโดยใช้กรณีการใช้งาน ซึ่งจะกลายเป็นมุมมองที่ห้า สถานการณ์จำลองอธิบายลำดับของการโต้ตอบระหว่างผู้กระทำ (Actor) และกระบวนการ (Use Case) ใช้เพื่อระบุองค์ประกอบทางสถาปัตยกรรม และตรวจสอบความถูกต้องของการออกแบบสถาปัตยกรรม นอกจากนี้ ยังใช้เป็นจุดเริ่มต้นสำหรับการทดสอบต้นแบบสถาปัตยกรรม มุมมองนี้เรียกอีกอย่างว่า มุมมองยูสเคส (Use Case) โดยใช้แผนภาพยูสเคส (Use Case Diagram)



รูปที่ 8.3 แบบจำลองมุมมองสถาปัตยกรรมแบบ 4 + 1 [16]



รูปที่ 8.4 แผนภาพยูสเคสที่สัมพันธ์กับแบบจำลองมุมมองสถาปัตยกรรมแบบ 4 + 1 [16]

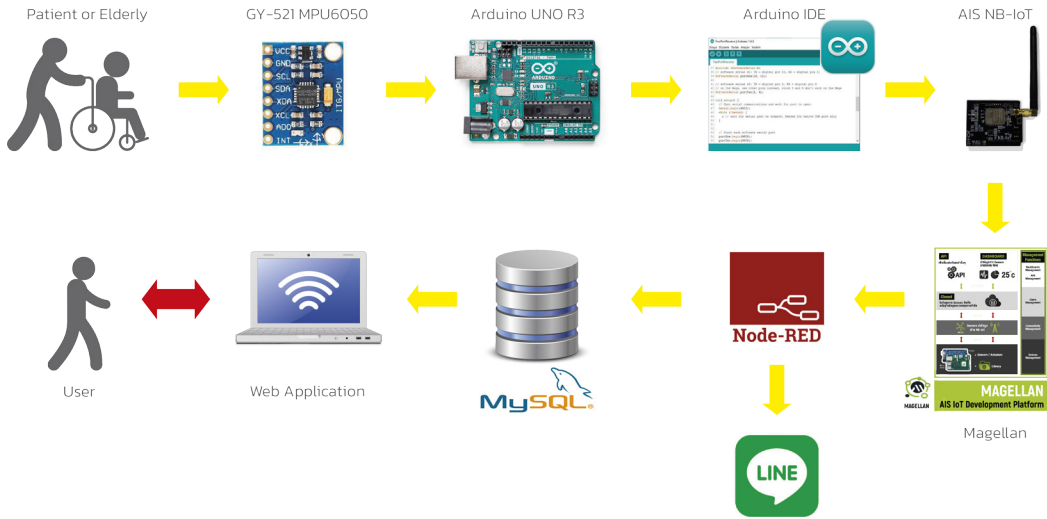


รูปที่ 8.10 สถาปัตยกรรมรับ-ให้บริการ (Client-Server) แบบ n-tiers [4]

แนวคิดของเว็บบริการ คือ เว็บที่สามารถทำงาน หรือให้บริการบางอย่างจากการร้องขอจากเซิร์ฟเวอร์ (Server) ต่างๆ ที่บูรณาการกันด้วยเทคโนโลยีเว็บบริการที่เชื่อมต่อแนวคิดการประมวลผลแบบกระจาย และการที่เว็บบริการมี UDDI (Universal Description, Discovery and Integration) ทำให้สามารถค้นหาบริการต่างๆ ที่ต้องการได้

8.2.3.4 สถาปัตยกรรมแบบท่อและการกรอง (Pipe and Filter architecture)

รูปแบบนี้ถูกนำมาใช้ตั้งแต่คอมพิวเตอร์ถูกใช้เป็นครั้งแรกเพื่อใช้การประมวลผลข้อมูล เมื่อการแปลงเป็นลำดับกับข้อมูลที่ประมวลผลในรูปแบบแบทช์ [15] สถาปัตยกรรมแบบท่อและการกรองนี้จะกลายเป็นวิธีการทำงานแบบเรียงลำดับ (Sequential) แบบกลุ่ม (Batch) การประมวลผลโดยมีการรวบรวมข้อมูลไว้ช่วงเวลาหนึ่งก่อนทำการประมวลผล การประมวลผลจะทำตามเวลาที่กำหนด ซึ่งอาจทำทุกวันหรือทุกสัปดาห์ ผู้ใช้ไม่สามารถเห็นผลลัพธ์ทันทีและไม่สามารถโต้ตอบกับระบบได้ สถาปัตยกรรมแบบนี้จะใช้กับการทำงานของอินเทอร์เน็ตประสานสรรพสิ่ง (Internet of Things, IoT) โดยมีการเก็บข้อมูลไว้ที่อุปกรณ์เซ็นเซอร์แล้วมีการส่งข้อมูลไปประมวลผลในเซิร์ฟเวอร์ [2] ดังแสดงในรูปที่ 8.11



รูปที่ 8.11 ตัวอย่างการใช้สถาปัตยกรรมแบบท่อและการรองรับระบบตรวจจัดการรถล้มของผู้ป่วย หรือผู้สูงอายุด้วยอินเทอร์เน็ตประสานรสวิ่ง [2]

8.2.4 สถาปัตยกรรมแอปพลิเคชัน (Application Architecture)

แอปพลิเคชันหรือซอฟต์แวร์ประยุกต์ เป็นการตอบสนองความต้องการซึ่งเน้นในการจัดการข้อมูล (Data) และสารสนเทศ (Information) เป็นหลัก

ข้อมูล (Data) หมายถึง ข้อเท็จจริงต่างๆ ที่ยังไม่ผ่านการประมวลผล ข้อมูลอาจอยู่ในรูปของตัวหนังสือ ตัวเลข ภาพ หรือเสียง [4]

สารสนเทศ (Information) หมายถึง ข้อมูลที่ผ่านการประมวลผลด้วยวิธีการต่างๆ เพื่อให้ได้ผลลัพธ์ตรงตามความต้องการ โดยพิจารณาใน 2 มิติ คือ มิติของผู้ใช้สารสนเทศ (User) และมิติของเวลา (Time) กล่าวคือ ในการประมวลผลนั้นต้องให้สอดคล้องกับผู้ใช้สารสนเทศ เพื่อที่จะนำไปใช้ประโยชน์ได้

สถาปัตยกรรมแอปพลิเคชันสามารถจัดกลุ่มได้ตามลักษณะการใช้งาน ซึ่งมีหลายรูปแบบคือ

- 1. ระบบประมวลผลธุรกรรม (Transaction Processing System)** หมายถึง ระบบที่ใช้ในการเปลี่ยนข้อมูลที่เกิดจากการปฏิบัติงานให้อยู่ในรูปแบบที่สามารถนำเข้าสู่อุปกรณ์คอมพิวเตอร์ เพื่อจัดเก็บรายละเอียดการทำธุรกรรม (Transaction) รวมถึงการประมวลผลและการแสดงผลรายงานที่เกิดขึ้นระหว่างการดำเนินการทางธุรกิจ การประมวลผลธุรกรรมจะควบคู่ไปกับการพัฒนาฐานข้อมูล (Database) สำหรับการใช้จัดเก็บข้อมูลที่เกิดจากการประมวลผลการทำธุรกรรมต่างๆ ซึ่งถือเป็นจุดเริ่มต้นของการนำเข้าข้อมูลในองค์กร ผู้ที่ใช้ฐานข้อมูลคือ พนักงาน เจ้าหน้าที่ ระดับปฏิบัติการ และเป็นงานประจำโดยดูที่ความถูกต้องของข้อมูลในส่วนที่ตัวเองรับผิดชอบเป็นหลัก

CHAPTER

9

การติดตั้งระบบและการบำรุงรักษา SYSTEM IMPLEMENTATION AND MAINTENANCE

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายรายละเอียดของการพัฒนาและติดตั้งระบบ
- เพื่ออธิบายวิธีการทดสอบซอฟต์แวร์แบบต่างๆ
- เพื่ออธิบายการถึงวิธีการติดตั้งระบบ
- เพื่ออธิบายการจัดทำเอกสาร
- เพื่ออธิบายการฝึกอบรม
- เพื่ออธิบายประเภทการบำรุงรักษาซอฟต์แวร์

เนื้อหาประจำบท (Contents)

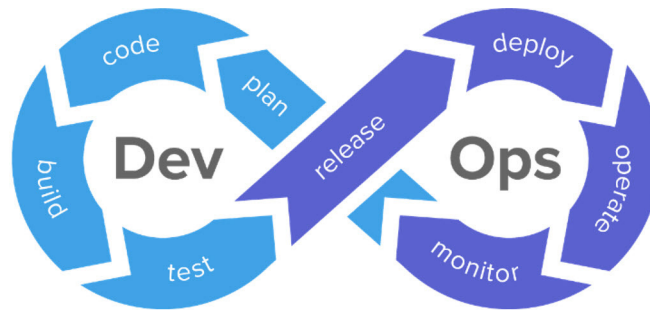
- 9.1 การพัฒนาและติดตั้งระบบ (System Implementation)
- 9.2 การทดสอบซอฟต์แวร์สำหรับแบบจำลองน้ำตก (Water Fall Model Software Testing)
- 9.3 การทดสอบซอฟต์แวร์สำหรับอไจล์ (Agile Software Testing)
- 9.4 การติดตั้งระบบ (System Installation)
- 9.5 การจัดทำเอกสาร (Documentation)
- 9.6 การฝึกอบรม (Training)
- 9.7 การบำรุงรักษาซอฟต์แวร์ (Software Maintenance)

9.1.3 วิธีการพัฒนาโปรแกรมแบบ Agile และ DevOps

การพัฒนาแบบดั้งเดิมหรือที่เรียกว่า แบบจำลองน้ำตก (Waterfall Model) ตามที่ได้กล่าวมาแล้วนั้น เหมาะกับระบบงานที่มีการพัฒนาขึ้นมาใหม่ซึ่งผ่านการวิเคราะห์ความต้องการมาอย่างดี ในขณะที่รูปแบบอไจล์ (Agile) คือแนวคิดในการทำงานที่ให้ความสำคัญในการสื่อสารกับผู้เกี่ยวข้องทุกฝ่าย เพื่อการปรับปรุงและพัฒนาผลิตภัณฑ์อยู่ตลอดเวลา เพื่อตอบสนองความต้องการของผู้ใช้งาน ซึ่งได้กล่าวมาแล้วในบทที่ 5

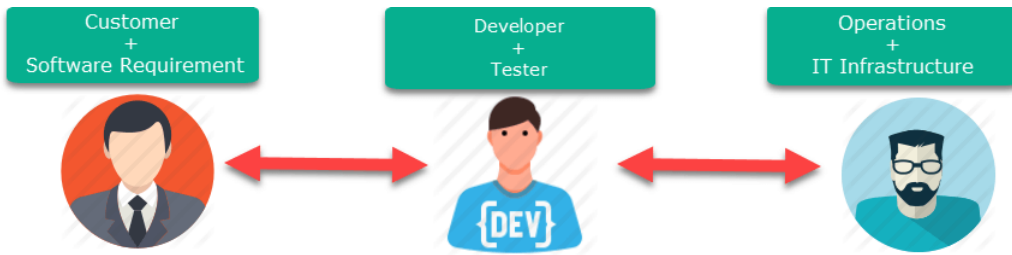
ในการพัฒนาแบบอไจล์นั้น ในบางครั้งหากมีการเปลี่ยนแปลงมากๆ หลังจากที่เราได้ดำเนินการไปแล้ว อาจทำให้เกิดปัญหาเรื่องของการดำเนินการ (Operation) เช่น ความเข้ากันได้ของระบบซึ่งอาจจะส่งผลไปยังโครงสร้างพื้นฐานที่เคยเตรียมไว้ และทำให้ทีมปฏิบัติการ (Operations) เองก็ไม่ทราบว่าจะตรวจสอบปัญหาได้จากที่ใด เพราะได้มีการเปลี่ยนแปลงที่มากเกินไป จึงเกิดแนวทางการทำงานที่ผสมผสานทั้งสองทีมเข้าด้วยกัน ระหว่างทีมพัฒนาซอฟต์แวร์ (Development, Dev) ที่ทำหน้าที่พัฒนาซอฟต์แวร์ที่สามารถตอบสนองความต้องการของลูกค้าได้อย่างรวดเร็ว กับทีมปฏิบัติการ (Operations, Ops) ที่ทำหน้าที่ให้บริการซอฟต์แวร์อย่างมีประสิทธิภาพ ทั้งสองทีมจะจับมือทำงานร่วมกันเป็นทีมเดียวกัน จึงเรียกว่า DevOps (Dev + Ops)

ประโยชน์ของโมเดล DevOps คือ ส่งมอบซอฟต์แวร์ได้รวดเร็วขึ้น ออกรุ่นใหม่ๆ ได้ดีขึ้น แก้ไขจุดบกพร่องได้เร็วขึ้น ปรับตัวต่อการเปลี่ยนแปลงของตลาดได้ดียิ่งขึ้น ลดต้นทุน ลดค่าใช้จ่าย และลดการหยุดทำงาน (Downtime) ทำให้ได้รับความพึงพอใจและความน่าเชื่อถือจากผู้ใช้มากขึ้น นอกจากนี้ยังสามารถใช้การเข้ารหัสการทดสอบ การจัดเตรียมโครงสร้างพื้นฐาน การปรับใช้และการตรวจสอบ และช่วยให้ย้อนกลับรหัสที่กำหนดเวอร์ชันได้ง่ายขึ้นในกรณีของการกู้คืนระบบ และทำให้สภาพแวดล้อมสามารถปรับขนาดได้ง่ายและปลอดภัย

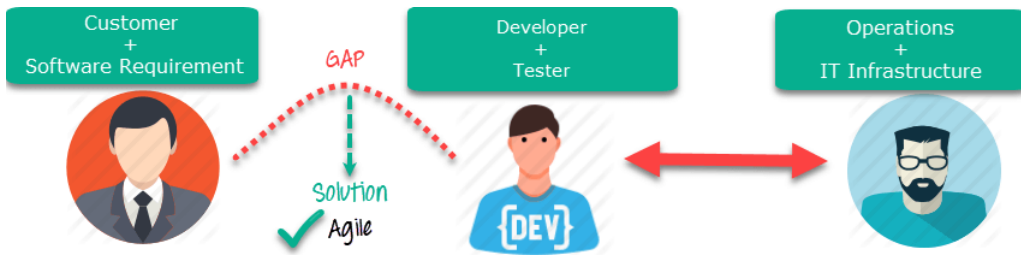


รูปที่ 9.11 กระบวนการทำงานแบบ DevOps [11]

จากรูปที่ 9.11 แสดงให้เห็นกระบวนการทำงานที่เชื่อมโยง DevOps เป็นชุดของการปฏิบัติที่บูรณาการการพัฒนาซอฟต์แวร์ และการดำเนินงานด้านเทคโนโลยีสารสนเทศ โดยมีจุดมุ่งหมายเพื่อย่นระยะเวลาการพัฒนาระบบ และให้การส่งมอบอย่างต่อเนื่องพร้อมกับคุณภาพซอฟต์แวร์ที่สูง ทั้งนี้ DevOps เป็นส่วนเสริมของการพัฒนาซอฟต์แวร์แบบอไจล์ [16]



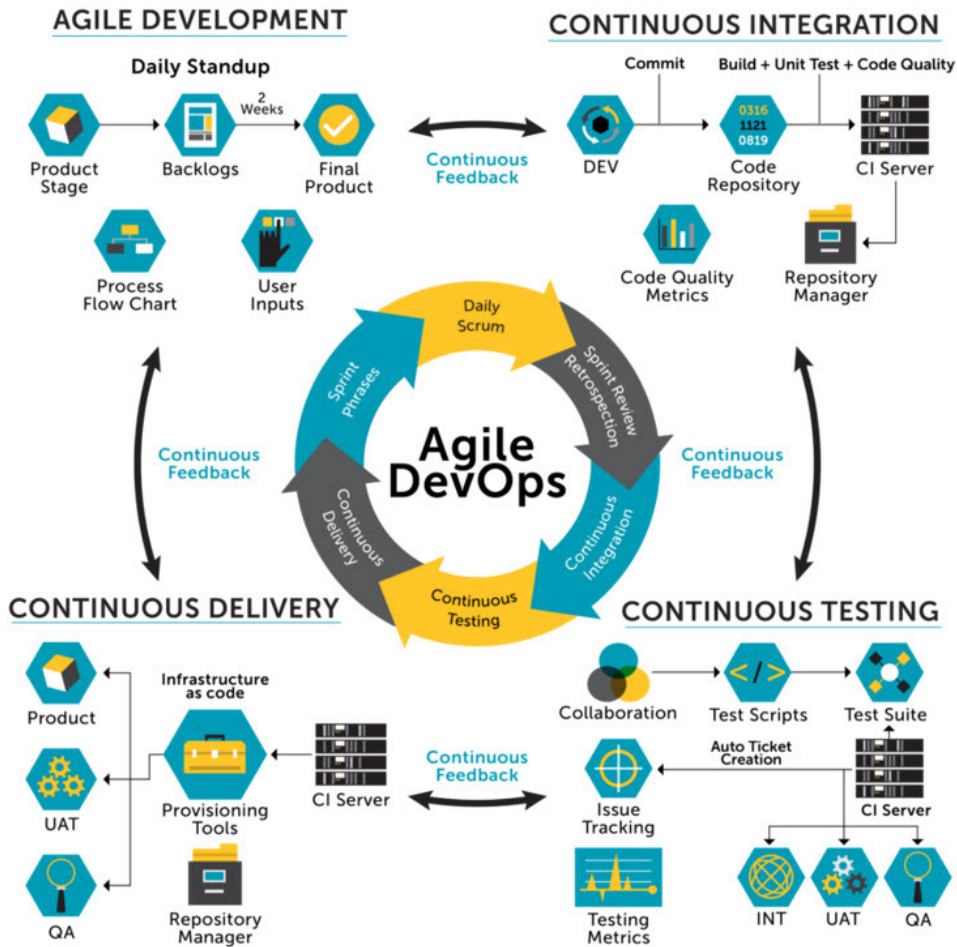
รูปที่ 9.12 แสดงการสื่อสารระหว่างผู้เกี่ยวข้องกับซอฟต์แวร์ (Stakeholders) ผู้ใช้หรือลูกค้า (Customer) ผู้พัฒนา (Developer, Dev) และผู้ดูแลระบบ (IT Operation) [9]



รูปที่ 9.13 Agile ช่วยจัดการกับช่องว่างในการสื่อสารของ Customer กับ Developer [9]



รูปที่ 9.14 DevOps ช่วยจัดการกับช่องว่างในการสื่อสารของ Developer และ IT Operations [9]



รูปที่ 9.15 การพัฒนาโดย Agile DevOps

(ที่มา : <https://www.pinterest.com/pin/164592561369080180/>)

- กระบวนการหรือการปฏิบัติของ DevOps เกี่ยวข้องกับชุดของกระบวนการทางเทคนิค เช่น การบูรณาการอย่างต่อเนื่อง (Continuous Integration, CI) การทดสอบอย่างต่อเนื่อง (Continuous Testing, CT) และการส่งมอบอย่างต่อเนื่อง (Continuous Delivery, CD)
- พื้นที่ที่สนใจให้ความสำคัญ จะมุ่งเน้นไปที่การรับประกันซอฟต์แวร์ที่มีคุณภาพในเวลาที่เหมาะสม การรับประกันคุณภาพจะทำได้โดยการตรวจสอบซอฟต์แวร์อย่างต่อเนื่องหลังจากการปรับใช้
- มีรอบระยะเวลาทั้งในส่วนของการเผยแพร่และการพัฒนา จะมุ่งเน้นไปที่ระยะการเผยแพร่ที่สั้นลง มุ่งให้มีการส่งมอบงานที่เร็วขึ้น แต่ติดตามผลการตอบรับอย่างต่อเนื่องทันที
- ผู้ให้ข้อเสนอแนะ จะมีการวัดผลจากทีมพัฒนาภายใน โดยอาจใช้เครื่องมือการตรวจสอบมาช่วยอย่างต่อเนื่อง

CHAPTER

10 วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส SERVICE ORIENTED SOFTWARE ENGINEERING

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายวิศวกรรมซอฟต์แวร์เชิงเซอร์วิส
- เพื่ออธิบายวิธีการบูรณาการระบบด้วยสถาปัตยกรรมเชิงเซอร์วิส
- เพื่ออธิบายเทคโนโลยีเว็บเซอร์วิส
- เพื่ออธิบายวิธีการแลกเปลี่ยนข้อมูลระหว่างแอปพลิเคชัน
- เพื่ออธิบายการพัฒนาาระบบด้วยสถาปัตยกรรมเชิงเซอร์วิส
- เพื่ออธิบายการพัฒนาาระบบด้วยสถาปัตยกรรมเชิงเซอร์วิส

เนื้อหาประจำบท (Contents)

- 10.1 วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส (Service Oriented Software Engineering)
- 10.2 รูปแบบการพัฒนาซอฟต์แวร์เชิงเซอร์วิส
- 10.3 เทคโนโลยีเว็บเซอร์วิส (Web Services)
- 10.4 รูปแบบข้อมูลที่ใช้ในเว็บเซอร์วิส
- 10.5 สถาปัตยกรรมเชิงเซอร์วิส (Service Oriented Architecture, SOA)
- 10.6 ไมโครเซอร์วิส (Microservices)
- 10.7 ซอฟต์แวร์คอนเทนเนอร์ (Software Container)

ตารางที่ 10.1 ความแตกต่างของวิศวกรรมซอฟต์แวร์เชิงเซอร์วิสและเชิงวัตถุ [7]

ประเด็นความแตกต่าง	วิศวกรรมซอฟต์แวร์เชิงเซอร์วิส	วิศวกรรมซอฟต์แวร์เชิงวัตถุ
ลักษณะนามธรรม (Abstract)	เซอร์วิส (Services)	วัตถุ (Objects)
ระดับความสัมพันธ์ของส่วนที่เกี่ยวข้องกัน (Coupling Level)	เป็นความสัมพันธ์แบบหลวมๆ (Loosely Coupling) เพราะแต่ละเซอร์วิสเป็นอิสระและไม่เกี่ยวข้องกันกับกระบวนการทางธุรกิจ	เป็นความสัมพันธ์แบบแน่น (Tightly Coupling) เพราะเป็นความสัมพันธ์ระหว่างคลาสในระบบเดียวกัน
จุดเน้น (Focus)	ระดับธุรกิจ	ระดับวัตถุ
ความซับซ้อน (Complexity)	สูง เพราะต้องเชื่อมโยงระบบที่มีสภาพแวดล้อมที่ต่างกันอย่างควบคุมได้ยาก	ปานกลางถึงสูง เพราะอยู่ในสภาพแวดล้อมเดียวกันที่ควบคุมได้
ระดับการทำงานร่วมกัน (Cohesion Level)	สูง	ปานกลาง
ความสามารถในการทำงานร่วมกัน (Interoperability)	สูง เพราะผู้ใช้สามารถเข้าถึงเซอร์วิสได้โดยไม่ต้องคำนึงถึงแพลตฟอร์ม (Platform) หรือภาษาในการพัฒนาระบบที่ต่างกัน	ต่ำ เพราะผู้ใช้อยู่ในแพลตฟอร์ม (Platform) หรือภาษาในการพัฒนาระบบเดียวกัน
การนำกลับมาใช้ใหม่ (Reusability)	สูง เพราะไม่จำเป็นต้องอยู่ในสภาพแวดล้อมเดิมๆ	ต่ำ เพราะต้องอยู่ในสภาพแวดล้อมเดิมเท่านั้น
ลักษณะการใช้งาน (Domain To Use)	การบูรณาการเชื่อมโยงกับระบบที่มีความแตกต่างกันหรือต่างองค์กร	ทำงานอยู่ในระบบเดียวกันหรือองค์กรเดียวกัน
ความรับผิดชอบของผู้ให้เซอร์วิส (Owner's Responsibility)	สูง เพราะผู้ให้เซอร์วิสต้องรับผิดชอบต่อการพัฒนา คุณภาพของเซอร์วิส การบำรุงรักษา การปรับใช้ การดำเนินการ และการจัดการ	ต่ำ เพราะผู้ใช้มีอิสระที่จะจัดการคลาสหรือวัตถุ และต้องรับผิดชอบด้วยตัวเอง
ความสามารถในการขยายตัว (Scalability)	สูง	ต่ำ
ส่วนต่อประสานแอปพลิเคชัน (Application Interface)	เน้นการอธิบายรายละเอียดของเซอร์วิส เช่น WSDL, SOAP, JSON	เน้นการอธิบายรายละเอียดของคลาสและวัตถุ

ระบบสารสนเทศภายในหน่วยงานที่มีการบูรณาการ จะสามารถประมวลเป็นสารสนเทศสำหรับผู้บริหาร (Executive Information System, EIS) ได้อย่างถูกต้อง และยังสามารถเชื่อมโยงไปยังซอฟต์แวร์ของหน่วยงานภายนอก เพื่อให้สามารถบูรณาการสารสนเทศกับหน่วยงานอื่นๆ ได้ นอกจากนี้ยังทำให้การบริการมีประสิทธิภาพในหลายช่องทาง เช่น ระบบการให้บริการ ณ จุดเดียว (Single Point Service) ระบบการชำระค่าธรรมเนียมด้วยระบบ e-Payment และระบบเว็บท่า (Portal) เป็นต้น โดยใช้ควบคู่กับการระบุตัวตน ดังรูป 10.3



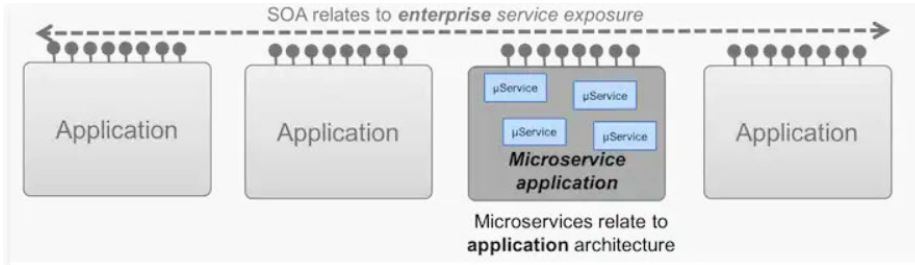
รูปที่ 10.3 แนวคิดของการออกแบบระบบสำหรับหน่วยงานภาครัฐและ/หรือภาคเอกชน [1]

(ปรับปรุงจาก : http://www.ops.moc.th/download/CIO_Board/MOC-Digital%20Plan.pdf)

จากรูป 10.3 แนวทางการพัฒนาระบบเทคโนโลยีดิจิทัลของกระทรวงต่างๆ ที่มุ่งเน้นประชาชนเป็นศูนย์กลาง (Citizen Centric) ซึ่งเป็นมุมมองการให้บริการเชิงรุกตอบสนองความต้องการของผู้ประกอบการ และประชาชนอย่างรู้ใจด้วยการนำเสนอรูปแบบ และบริการที่ทันสมัยหลากหลายช่องทาง (Channel) โดยสร้างความร่วมมือกับหน่วยงานระดับกรมในสังกัดกระทรวงพาณิชย์ในการใช้ข้อมูล (Data Source) การพัฒนาโครงสร้างพื้นฐาน (Supporting Infrastructure) การพัฒนาระบบให้บริการ (Front Office) การพัฒนาระบบบริหารจัดการภายใน (Back Office) รวมทั้งการนำเสนอและส่งมอบข้อมูล/บริการ (Delivery & Presentation) ร่วมกัน

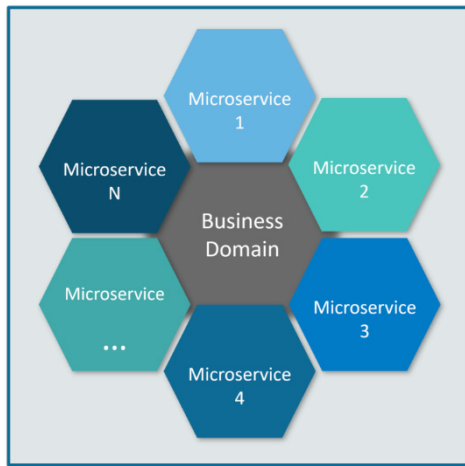


ความแตกต่างที่สำคัญระหว่างทั้งสองแนวทางขึ้นอยู่กับขอบเขตของการพัฒนา กล่าวคือ สถาปัตยกรรมเชิงเซอร์วิส (SOA) เป็นการพัฒนาระดับองค์กรตามที่ได้อธิบายมาแล้วข้างต้น ในขณะที่สถาปัตยกรรมไมโครเซอร์วิส (Microservices) เป็นการพัฒนาระดับแอปพลิเคชัน แม้ว่าทั้งสองสถาปัตยกรรมทั้งสองแบบนี้จะมีความเป็นโมดูลของแอปพลิเคชันเหมือนกัน แต่ก็มีความแตกต่างกันในวิธีการปรับใช้เซอร์วิสและขนาดของโมดูล สถาปัตยกรรมไมโครเซอร์วิสโดยละเอียด ดังนี้



รูปที่ 10.26 ความสัมพันธ์ของไมโครเซอร์วิสกับสถาปัตยกรรมเชิงเซอร์วิส

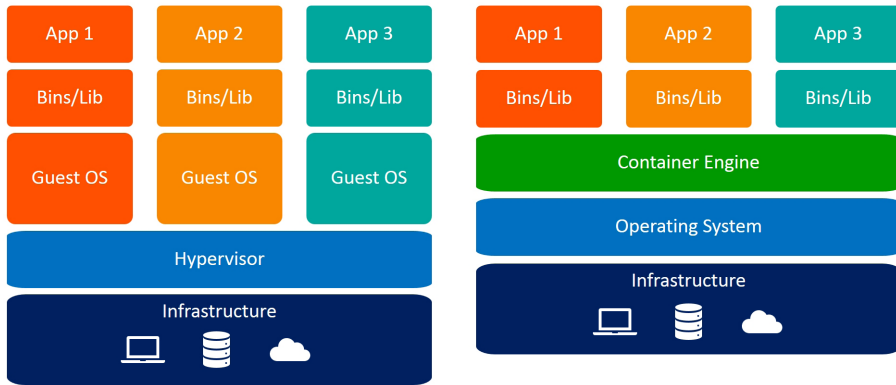
(ที่มา : <https://www.ibm.com/cloud/blog/soa-vs-microservices>)



รูปที่ 10.27 ภาพรวมของแนวคิดในการกระจายการทำงานแบบไมโครเซอร์วิส (Microservices)

(ที่มา : <https://www.edureka.co/blog/microservices-security>)

ไมโครเซอร์วิส เป็นวิธีการแยกส่วนของซอฟต์แวร์ ที่มีจุดมุ่งหมายเพื่อแบ่งระบบซอฟต์แวร์ขนาดใหญ่ออกเป็นส่วนประกอบขนาดเล็ก ด้วยสถาปัตยกรรมไมโครเซอร์วิส แอปพลิเคชันถูกสร้างขึ้นด้วยกลุ่มคอมโพเนนต์อิสระที่รันแต่ละกระบวนการแอปพลิเคชันเป็นเซอร์วิส สถาปัตยกรรมดังกล่าวช่วยให้แอปพลิเคชันปรับขนาดได้ง่ายขึ้น เร่งกระบวนการพัฒนามีพื้นที่สำหรับการทดลอง และลดเวลาออกสู่ตลาดสำหรับคุณสมบัติใหม่ๆ



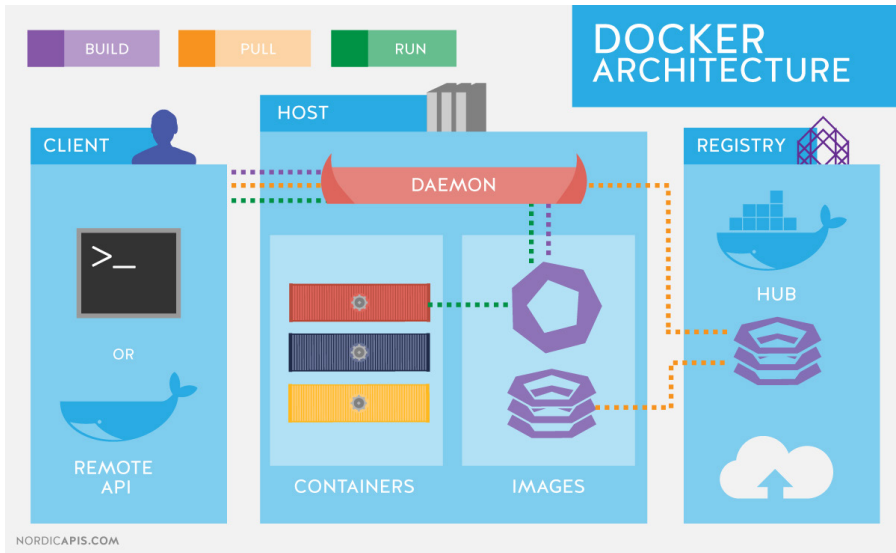
Virtual Machines

Containers

รูปที่ 10.29 การเปรียบเทียบความแตกต่างระหว่าง Virtual Machine กับ Containers

(ที่มา : <https://www.docker.com/what-docker>)

ซอฟต์แวร์คอนเทนเนอร์ (Software Container) เป็นแนวคิดของการสร้างสภาพแวดล้อมเฉพาะให้ซอฟต์แวร์ทำงานได้โดยไม่ส่งผลกระทบต่อซอฟต์แวร์ตัวอื่นในระบบปฏิบัติการเดียวกัน ผู้พัฒนาสามารถนำซอฟต์แวร์คอนเทนเนอร์ไปทำงานในคอมพิวเตอร์หรือเซิร์ฟเวอร์ (Server) เครื่องใดๆ ก็ได้ โดยโปรแกรมที่ถูกบรรจุอยู่ยังคงทำงานได้เหมือนเดิม ตัวจัดการคอนเทนเนอร์ที่นิยมในกลุ่มผู้พัฒนาซอฟต์แวร์มีชื่อว่า “ด็อกเกอร์” (Docker)



รูปที่ 10.30 องค์ประกอบของด็อกเกอร์ (Docker)

(ที่มา : <https://thingsolver.com/hello-docker/>)

CHAPTER

11

การพัฒนาบุคลากรด้านดิจิทัล DIGITAL HUMAN RESOURCE DEVELOPMENTS

วัตถุประสงค์ (Objectives)

- เพื่ออธิบายกระบวนการพัฒนาบุคลากรด้านดิจิทัล
- เพื่ออธิบายทักษะดิจิทัลที่จำเป็นต่อการปรับเปลี่ยนสู่ดิจิทัล
- เพื่ออธิบายการพัฒนาบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล

เนื้อหาประจำบท (Contents)

- 11.1 ทักษะด้านดิจิทัล
- 11.2 สมรรถนะของบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล
- 11.3 การพัฒนาบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล

- **มิติที่ 5 ใช้ดิจิทัลเพื่อขับเคลื่อนการเปลี่ยนแปลงและสร้างสรรค์** ประกอบด้วย 1 กลุ่มทักษะ ได้แก่ กลุ่มทักษะด้านการขับเคลื่อนการเปลี่ยนแปลงด้านดิจิทัล (Digital Transformation Skill Set)

ยุทธศาสตร์ด้านเทคโนโลยีดิจิทัล ต้องให้ความสำคัญในการพัฒนาบุคลากรด้านเทคโนโลยีดิจิทัลในทุกระดับ การพัฒนากำลังคนด้านเทคโนโลยี และบุคคลทั่วไปให้มีความสามารถในการสร้างสรรค์ ผลิตภัณฑ์ และใช้สารสนเทศอย่างมี วิจารณญาณและรู้เท่าทัน (Information Literacy) โดยเน้นการสร้างบุคลากรทักษะสูง (Highly Skilled Professionals) โดยมีเป้าหมายคือ บุคลากรทั้งภาครัฐและเอกชนสามารถเข้าถึงและนำ IT มาใช้ประโยชน์ในการทำงาน และการเรียนรู้ ซึ่งมีมาตรการสำหรับการพัฒนาบุคลากร ดังนี้

1. กำหนดมาตรฐานความรู้ IT ทุกระดับ ตำแหน่ง และมีกลไกตามมาตรฐาน
2. ส่งเสริมการพัฒนาความรู้ด้าน IT
3. สร้างแรงจูงใจรวมถึงความก้าวหน้าในหน้าที่การงาน (Career Path) ที่เหมาะสม

ทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ

เพื่อการปรับเปลี่ยนเป็นรัฐบาลดิจิทัล

ที่มา

คณะรัฐมนตรี ในการประชุมเมื่อวันที่ 26 กันยายน 2560 มอบหมายให้สถาบันคุณวุฒิวิชาชีพ (องค์การมหาชน) ร่วมกับสำนักงาน ก.พ. จัดทำรายละเอียดทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ

สำนักงาน ก.พ.

วัตถุประสงค์

- 📍 เพื่อสนับสนุนการปรับเปลี่ยนภาครัฐเป็นรัฐบาลดิจิทัล
- 📍 เพื่อให้ข้าราชการและบุคลากรภาครัฐมีทักษะด้านดิจิทัล เป็นแนวทางชัดเจน
- 📍 เพื่อให้ส่วนราชการและหน่วยงานของรัฐ มีเครื่องมือในการพัฒนาบุคลากรให้มีทักษะดิจิทัลที่เหมาะสม

เป้าหมาย / ความคาดหวัง

Open and Connected Government

รัฐบาลที่มีภาคีเชื่อมโยงการทำงานและเปิดและเชิญชวน การมีส่วนร่วมของประชาชน

Smart Government for Citizen

รัฐบาลที่มีรูปแบบการทำงาน ที่มีประสิทธิภาพและตรงกับ ความต้องการของประชาชน

Digital Culture

รัฐบาลที่มีวัฒนธรรมองค์กร ดิจิทัล ใช้เทคโนโลยีการตัดสินใจ และนวัตกรรมเชิงรุก

องค์ประกอบ

ทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ ประกอบด้วย 4 องค์ประกอบ ดังนี้

39

ความสามารถ

23

ความรู้

18

ประสบการณ์

5

คุณลักษณะ

+

5

สมรรถนะ (5/27/2552)

Foundation

- ความสามารถด้านความเข้าใจและใช้เทคโนโลยีดิจิทัล (Digital Literacy)
- ความสามารถด้านความรู้คู่กับและการปฏิบัติตามกฎหมาย นโยบายและมาตรฐานการจัดการ (Digital Governance, Standard, and Compliance)

Technology & Design

- ความสามารถด้านเทคโนโลยีดิจิทัลเพื่อยกระดับศักยภาพองค์กร (Digital Technology)
- ความสามารถด้านการออกแบบกระบวนการและการให้บริการด้วยระบบดิจิทัล (Digital Process and Service Design)

Management

- ความสามารถด้านการบริหารกลยุทธ์และการจัดการโครงการ (Strategic and Project Management)
- ความสามารถด้านผู้นำดิจิทัล (Digital Leadership)
- ความสามารถด้านการขับเคลื่อนการเปลี่ยนแปลงด้านดิจิทัล (Digital Transformation)

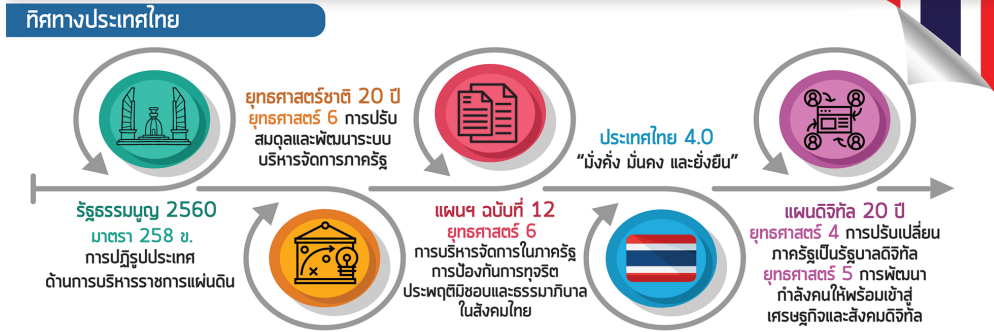
รูปที่ 11.1 องค์ประกอบทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ [3]

Book.indb 351

14/3/2565 BE 17:08



แนวทางพัฒนา ทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ เพื่อการปรับเปลี่ยนเป็นรัฐบาลดิจิทัล (มติคณะรัฐมนตรี วันที่ 26 กันยายน พ.ศ. 2560)



การพิจารณาของคณะกรรมการข้าราชการพลเรือน

- เมื่อวันที่ 16 มกราคม 2560 "ปรับเปลี่ยนภาครัฐเป็นรัฐบาลดิจิทัล" เป็นประเด็นพัฒนาสำคัญ
- เมื่อวันที่ 12 มิถุนายน 2560 เห็นชอบร่างแนวทางพัฒนาทักษะดิจิทัลฯ และนำไปปรับใช้

การปรับเปลี่ยนภาครัฐเป็นรัฐบาลดิจิทัล

หัวใจสำคัญของการขับเคลื่อนการพัฒนาประสิทธิภาพการบริหารจัดการภาครัฐ คือ **"บุคลากรภาครัฐ"** และเพื่อขับเคลื่อนให้เกิด **"การปรับเปลี่ยนภาครัฐเป็นรัฐบาลดิจิทัล"** จึงเห็นควรนำเรื่อง **"การสร้างและพัฒนามนุษย์ภาครัฐ"** มาเป็นกลไกสนับสนุนการขับเคลื่อนการเปลี่ยนแปลงที่สำคัญ

แนวทางการพัฒนาทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ (พ.ศ. 2561 - 2565)

วัตถุประสงค์ เพื่อให้ภาครัฐมีกำลังคนที่มีทักษะด้านดิจิทัลที่เหมาะสม ข้าราชการและบุคลากรภาครัฐสามารถปรับตัวให้เท่าทันกับการเปลี่ยนแปลงด้านเทคโนโลยี และมีความพร้อมที่จะปฏิบัติงานตามบทบาทและพฤติกรรมที่คาดหวัง

เป้าหมาย ภาครัฐใช้เทคโนโลยีดิจิทัลในการยกระดับคุณภาพการบริหารจัดการและการบริการเพื่ออำนวยความสะดวกให้ประชาชนและผู้รับบริการเพื่อความเท่าเทียมและลดความเหลื่อมล้ำ

ภายในปี พ.ศ. 2565 ข้าราชการและบุคลากรภาครัฐ สามารถปรับตัวมีทักษะและศักยภาพที่เหมาะสมต่อการปรับเปลี่ยนภาครัฐเป็นรัฐบาลดิจิทัล

ผู้บริหารระดับสูง เป็นผู้นำด้านดิจิทัลภาครัฐ ที่สามารถกำหนดนโยบายและทิศทางขององค์กร รวมถึงกระตุ้นและผลักดันให้ข้าราชการและบุคลากรภาครัฐรวมถึงหน่วยงานที่เกี่ยวข้อง ปรับเปลี่ยนรูปแบบการดำเนินงาน หรือการให้บริการองค์กร โดยนำเทคโนโลยีดิจิทัลมาใช้

ผู้อำนวยการกอง เป็นผู้ริเริ่มการเปลี่ยนแปลงด้านดิจิทัลระดับองค์กร ที่สามารถสื่อสารนโยบายขององค์กรสู่ระดับปฏิบัติ หรือสั่งการ กำหนดแนวทาง วางแผน กำหนด ติดตามและให้การสนับสนุนรูปแบบการดำเนินงานให้อยู่ในรูปแบบดิจิทัล

ผู้ทำงานด้านนโยบายและงานวิชาการ เป็นผู้ใช้ข้อมูลดิจิทัลเพื่อสนับสนุนนโยบายที่สามารถคิด วิเคราะห์ สังเคราะห์ และใช้ข้อมูลและเทคโนโลยีดิจิทัล

ผู้ทำงานด้านบริการ เป็นผู้อำนวยความสะดวกด้านดิจิทัลภาครัฐที่สามารถให้บริการ อำนวยความสะดวกให้แก่ประชาชนและผู้รับบริการ

ผู้ปฏิบัติงานเฉพาะด้านเทคโนโลยีดิจิทัล เป็นผู้ปรับเปลี่ยนเทคโนโลยีขององค์กรที่สามารถบริหารโครงการหรือเลือกเทคโนโลยีที่เหมาะสม

ผู้ปฏิบัติงานทั่วไป เป็นผู้ปฏิบัติงานภาครัฐที่รู้เท่าทันการเปลี่ยนแปลงของเทคโนโลยี และสามารถใช้ประโยชน์จากเทคโนโลยีได้อย่างเหมาะสม ถูกต้อง และปลอดภัย

ปัจจัยชี้วัดความสำเร็จ

Digital Government
เป็นองค์กรที่สร้างสรรค์นวัตกรรมโดยนำเทคโนโลยีดิจิทัลมาใช้

Connected Government
มีการเชื่อมโยงระหว่างภาครัฐ เอกชน และประชาชน

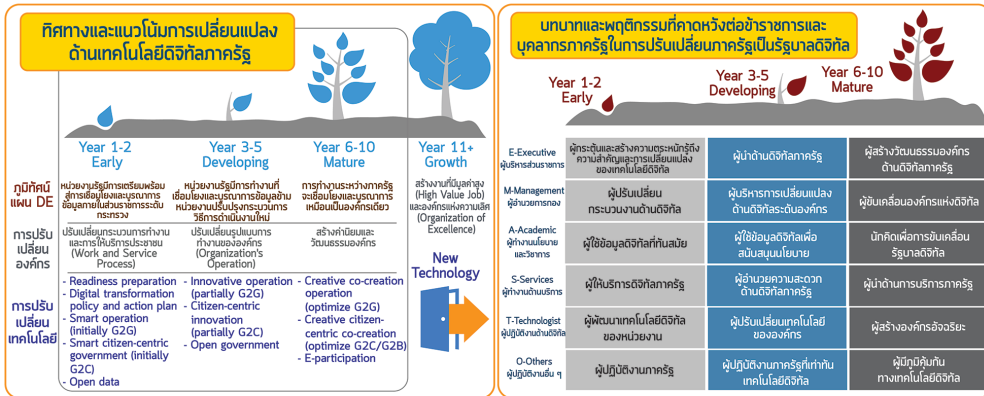
Open Government
มีการเปิดเผยข้อมูล ที่สนับสนุนการสร้างรัฐบาลแบบเปิด

ผลที่คาดว่าจะได้รับจากการพัฒนาทักษะด้านดิจิทัล

Global Indicators

E-Government Ranking (United Nations) 25% ปี 59 อันดับ 77/193 (0.5522 คะแนน)	The Networked Readiness Index (World Economic Forum) 55 ปี 59 อันดับ 62/139 (4.2 คะแนน)	Global Competitiveness (World Economic Forum) 25 ปี 59 อันดับ 34/138 (คะแนน 4.64/7)
Global Open Data Index (Open Knowledge International) 30 ปี 58 อันดับ 42/122 (ร้อยละ 39)		Ease of Doing Business (The World Bank) 40 ปี 60 อันดับ 49/190 (ร้อยละ 71.42)

รูปที่ 11.2 แนวทางการพัฒนาทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ [3]



ทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ

<p>มคอที่ 1 รู้เท่าทันและใช้เทคโนโลยีเป็น</p> <p>Digital Literacy</p>	<p>มคอที่ 2 เข้าใจนโยบาย กฎหมายและมาตรฐาน</p> <p>Digital Governance, Standard and Compliance</p>	<p>มคอที่ 3 ใช้ดิจิทัลเพื่อการประยุกต์และพัฒนา</p> <p>Digital Technology Digital Process and Service Design and Assurance</p>	<p>มคอที่ 4 ใช้ดิจิทัลเพื่อการวางแผนบริหารจัดการและนำองค์กร</p> <p>Project and Strategic Management Digital Leadership</p>	<p>มคอที่ 5 ใช้ดิจิทัลเพื่อขับเคลื่อนการเปลี่ยนแปลงและสร้างสรรค์</p> <p>Digital Transformation</p>
--	---	--	---	---

แนวทางพัฒนากำลังคนภาครัฐให้มีทักษะด้านดิจิทัล

- กำหนดให้ "การพัฒนาคนเพื่อสร้างและพัฒนาระบบดิจิทัล" และ "การสร้างและพัฒนาคนให้เท่าทัน และสามารถใช้อุปกรณ์อย่างรอบรู้" ซึ่งเป็นประเด็นหลักในการพัฒนา
- ให้มีการจัดการพัฒนาคนแบบบูรณาการ โดยนำของ ผู้บริหาร พ.อ.กอง ผู้ปฏิบัติงานและนักไอที มาเรียนรู้และเติมเต็มซึ่งกันและกัน เพื่อสร้างระบบดิจิทัลของหน่วยงาน
- กำหนดให้เป็นหน้าที่ของบุคลากรภาครัฐในการพัฒนาตนเอง และนำวิธีการพัฒนาแบบ 70 : 20 : 10 มาใช้ (70 พัฒนาตนเองและเรียนรู้จากการปฏิบัติงาน : 20 เรียนรู้จากผู้อื่นและการสอนงาน : 10 เรียนรู้จากการฝึกอบรม)
- ให้มีการปรับรูปแบบการพัฒนา โดยนำแนวทางจัดการเรียนรู้แบบผสมผสานมาใช้ และในการฝึกอบรมให้ลดบรรยาย และเพิ่มการเรียนรู้แบบอื่นในสัดส่วน 60 : 40

แนวทางการขับเคลื่อนการพัฒนา

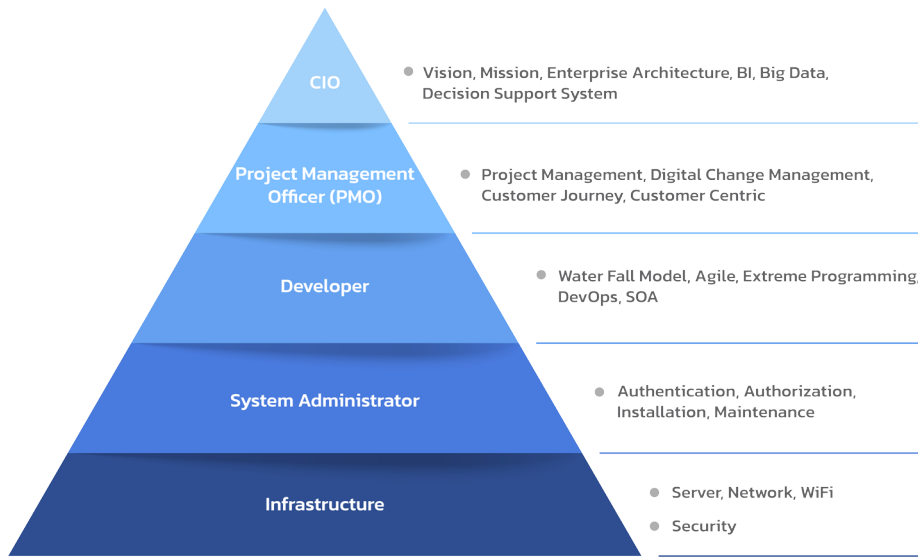
<p>การกำหนดผู้รับผิดชอบการขับเคลื่อนการพัฒนาที่ชัดเจน</p> <p>กระทรวงดิจิทัลฯ ให้การส่งเสริม สนับสนุน และดำเนินการเพื่อให้ข้าราชการและบุคลากรภาครัฐได้รับการพัฒนาทักษะดิจิทัล</p> <p>CEO และ CIO นำแนวทางพัฒนาทักษะด้านดิจิทัลไปดำเนินการให้บรรลุผลสำเร็จในระดับหน่วยงาน</p>	<p>การวางแนวทางการดำเนินงานเพื่อพัฒนาทักษะด้านดิจิทัล</p> <p>ให้หน่วยงานของรัฐประสานและทำงานแบบบูรณาการร่วมกับกระทรวงดิจิทัลฯ และสำนักงาน ก.พ.</p> <p>จัดเตรียมงบประมาณสำหรับการพัฒนาและการประเมินทักษะด้านดิจิทัล</p>
<p>การจัดทำรายละเอียดทักษะและการประเมิน</p> <p>ให้สำนักงาน ก.พ. และสถาบันคุณวุฒิวิชาชีพ (องค์การมหาชน) รับผิดชอบดำเนินการ และนำเสนอ ก.พ. พิจารณาประกาศใช้ต่อไป</p>	<p>การติดตามการดำเนินงาน</p> <p>กำหนดให้ "การพัฒนาเพื่อการปรับเปลี่ยนเป็นรัฐบาลดิจิทัล" เป็นหนึ่งในปัจจัยที่นำมาพิจารณาประเมินบุคคลและองค์กร</p>
<p>การพัฒนากำลังคนด้านดิจิทัลภาครัฐ</p> <p>ให้กระทรวงดิจิทัลฯ ร่วมกันกับสำนักงาน ก.พ. พัฒนาการบริหารกำลังคนภาครัฐสำหรับผู้ปฏิบัติงานเทคโนโลยีดิจิทัลภาครัฐ</p> <p>ให้นำพนักงานราชการที่มีศักยภาพสูงมาเป็นอัตรากำลังเสริมระยะสั้น</p>	<p>การประเมินข้าราชการประเภทบริหารระดับสูง</p> <p>(ปลัดกระทรวง อธิบดี เอกอัครราชทูต ผู้ว่าราชการจังหวัด)</p> <p>ดำเนินการตามบทบาทและพฤติกรรมที่คาดหวังในการปรับเปลี่ยนเป็นรัฐบาลดิจิทัล</p>
<p>สำนักงาน ก.พ.</p> <p>การประเมินข้าราชการประเภทบริหารระดับสูง</p>	<p>นพส</p> <p>การประเมินส่วนราชการองค์กรประเภทที่ 4 (Innovation Base)</p> <p>การนำวัฒนธรรมการทำงานหรือการให้บริการให้ริเริ่มด้วยเทคโนโลยีดิจิทัลมาปรับปรุงหรือพัฒนาการเชื่อมโยงข้อมูลโดยใช้เทคโนโลยีดิจิทัล การเปิดเผยข้อมูลภาครัฐเพื่อสร้างรัฐบาลแบบเปิด</p>

รูปที่ 11.3 แนวทางพัฒนาทักษะด้านดิจิทัลของข้าราชการและบุคลากรภาครัฐ (ต่อ) [3]



11.2 สมรรถนะของบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล

สมรรถนะของบุคลากรที่เกี่ยวข้องกับการพัฒนาเทคโนโลยีดิจิทัล ซึ่งปฏิบัติงานในหน่วยงานที่พัฒนาและดูแลระบบเทคโนโลยีสารสนเทศ (IT Department) สามารถแบ่งเป็น 5 ระดับ ดังแสดงตามรูปที่ 11.4 [2]



รูปที่ 11.4 ระดับของกลุ่มบุคลากรด้านการพัฒนาเทคโนโลยีดิจิทัล

กลุ่มที่ 1 ระดับประธานฝ่ายสารสนเทศ (Chief Information Officer, CIO) คือ ผู้บริหารสูงสุดในองค์กรในด้านเทคโนโลยีสารสนเทศ มีหน้าที่ในการตัดสินใจ กำหนดทิศทางนโยบายการบริหารด้านเทคโนโลยีสารสนเทศขององค์กร รวมถึงการควบคุมดูแลบริหารของหน่วยงานด้านเทคโนโลยีดิจิทัลในภาพรวมทั้งหมด ซึ่งควรจะมีความรู้ในด้านต่อไปนี้ [2]

- เข้าใจในศักยภาพและความเสี่ยงในการใช้เทคโนโลยีสารสนเทศ
- เข้าใจในการใช้เทคโนโลยีสารสนเทศเป็นกลยุทธ์ในการขับเคลื่อนยุทธศาสตร์ของหน่วยงาน
- แปลงแผนสู่การปฏิบัติอย่างเป็นรูปธรรม
- เข้าใจวิธีการจัดการกระบวนการทำงาน การติดตาม และประเมินผลระบบเทคโนโลยีสารสนเทศ
- เข้าใจกฎหมายด้านเทคโนโลยีสารสนเทศ
- เข้าใจกระบวนการวางแผน บริหารจัดการ โครงการเทคโนโลยีสารสนเทศ และการจัดการผู้รับจ้าง (Outsourcing Management)
- เข้าใจการเจรจาต่อรอง และสามารถสื่อสารกับผู้พัฒนาเทคโนโลยีสารสนเทศ